
数学之美
吴军 著
JUST PUB
[redacted]
人民邮电出版社
POSTS & TELECOM PRESS



我大学的专业是计算数学，但读到吴军老师的“数学之美”系列文章，才发现马尔可夫链、矩阵计算，甚至余弦函数原来都如此亲切，并且栩栩如生；才发现自然语言和信息处理这么有趣；才真正明白“数学是科学的皇后”这句名言。相信认真读完这本《数学之美》的朋友们，算法功力都会暴涨N倍，更重要的是发现了数学背后的无穷魅力，学会欣赏数学之美。

——**蒋涛** / CSDN＆《程序员》创始人

最初看到“数学之美”，是谷歌黑板报上的连载文章。里面的公式并不是很多，但是很多看似颇为复杂的概念，吴军老师却能够如讲故事般娓娓道出，着实看出作者对这些问题有着深入且独到的见解，读后受益匪浅。这次有幸在《数学之美》出版之前拜读了初稿，欣喜看到新书在章节连贯和语言方面都较黑板报的连载文章有了较大的提高，相信每一个喜欢数学、乐意欣赏数学之美的读者，一定会觉得开卷有益。

——**张磊** / 微软亚洲研究院主管研究员

我不做研究，也自觉没有做研究的底子。然而，数年前看到吴军老师的“数学之美”系列时仍然还是被深深地迷住了。正如作为一个十几年的科幻爱好者，深信在平凡的生活和工作之余应得闲仰望星空一样，作为生活在信息社会的个体，在上微博、搜Google、发邮件之余，关上显示器，能够透过《数学之美》这样的杰作，一窥纷繁涌动的数字世界背后的引擎——数学之美，实乃一件幸事。

——**刘未鹏** / 《暗时间》作者

第一次接触吴军老师的“数学之美”系列，是在搜索bloom filter资料时，读了其中一篇后，就把其他的文章都读了，感触很多：首先，改变了观点，原以为在计算机系学到的数学基础在工作中一无是处，现在懂得，知识要落地，最重要的是理解知识的由来；其次，任何复杂的问题最终可以用简单的方式去解决，我们往往会陷入不断给问题增加难度的复杂解法，而忽视了简单直接有效的方法。

“数学之美”系列文章，整体和细节的度把握得很好，通过具体的例子让读者学到的是思考问题的方式，同时留了很多问题给愿意钻研的人做进一步深入思考。BTW，“数学之美”系列，是我在技术领域介绍中读过的最好的文章之一，让人学会如何化繁为简，如何用数学去解决工程问题，如何跳出固有思维不断去思考创新。

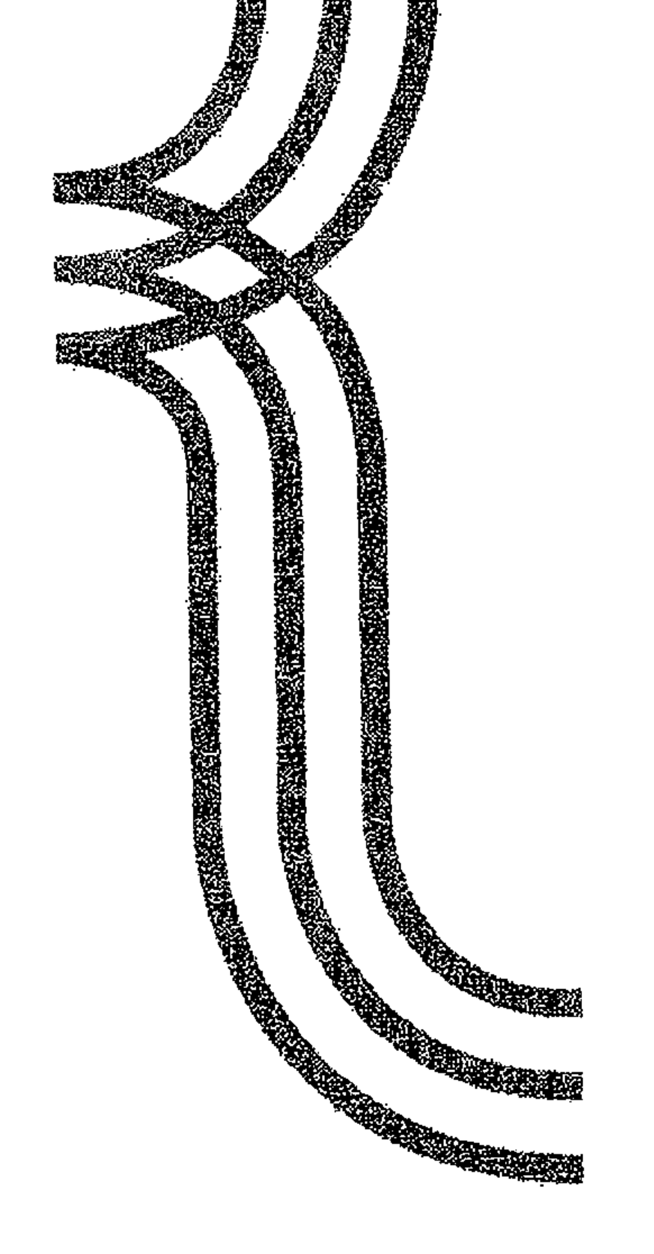
——**岑文初** / 淘宝开放平台技术产品负责人

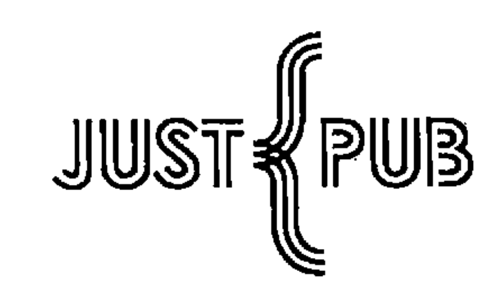


审稿编辑：李琳骁
责任编辑：俞　彬
策划编辑：周　筠



ISBN 978-7-115-28282-8
定价：45.00 元





数学之美

Beauty of Mathematics

吴军　著

人民邮电出版社
北京

图书在版编目（CIP）数据

数学之美 / 吴军著. — 北京 : 人民邮电出版社,
2012.6(2012.6 重印)
ISBN 978-7-115-28282-8

Ⅰ. ①数… Ⅱ. ①吴… Ⅲ. ①电子计算机—数学基础
Ⅳ. ①TP301.6

中国版本图书馆CIP数据核字(2012)第088566号

内 容 提 要

几年前，“数学之美”系列文章原刊载于谷歌黑板报，获得上百万次点击，得到读者高度评价。读者说，读了“数学之美”，才发现大学时学的数学知识，比如马尔可夫链、矩阵计算，甚至余弦函数原来都如此亲切，并且栩栩如生，才发现自然语言和信息处理这么有趣。

今年，作者吴军博士几乎把所有文章都重写了一遍，为的是把高深的数学原理讲得更加通俗易懂，让非专业读者也能领略数学的魅力。读者通过具体的例子学到的是思考问题的方式——如何化繁为简，如何用数学去解决工程问题，如何跳出固有思维不断去思考创新。

数 学 之 美

◆ 著　　　　吴　军
　责任编辑　俞　彬
　审稿编辑　李琳骁
　策划编辑　周　筠
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
　邮编　100061　电子邮件　315@ptpress.com.cn
　网址　http://www.ptpress.com.cn
　北京铭成印刷有限公司印刷
◆ 开本：720×960　1/16
　印张：19　彩插：1
　字数：248 千字　2012 年 6 月第 1 版
　印数：40 001 – 80 000 册　2012 年 6 月北京第 4 次印刷

ISBN 978-7-115-28282-8

定价：45.00 元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

本书谨献给我的家人。

愿科学之精神在国民中得到普及，愿中国年轻的一代涌现更多的杰出专业人才。

出版说明

“数学之美”最初是从 2006 年起在 Google 中国的官方博客 —— 谷歌黑板报上连载的系列博客。当时我写这个系列的原因完全是应原黑板报版主吴丹丹女士（现任职于苹果公司）之请，希望介绍一点 Google 的技术，盛情难却，便勉为其难接下了这个任务。这个任务的难处在于既要介绍 Google 的技术，又不能泄密。于是我只好采用了仅介绍基本原理尤其是数学原理的方法来写文章。加上我自己对数学比较感兴趣，博士论文也是以数学为主的题目，因此，便写成了介绍我所从事的信息处理领域的数学基础的系列文章。当初我并没有计划写多少篇，只打算有空就抽时间写一点，写到哪儿算哪儿，没时间写就算了。不想刊登了几篇后，受到 IT 行业广大从业人员的关注和喜爱。在互联网上被转载了上万次，读者有上百万。大家都鼓励我写下去，于是便陆陆续续写了 20 多篇。

后来我在 Google 的工作越来越多，同时在公司外还有很多要履行的义务，便很少有精力再写这个系列了。2009 年，李开复离开 Google，谷歌黑板报的第二任版主崔瑾女士也随他去了创新工场（后来转到豌豆荚）。Google 再也没有人敦促我继续为谷歌黑板报写博客了。令我感动的是，这么多年后，还不断有读者关注这个系列，并且时不时地问我是否能把这个系列写完，是否可能出书。因此，从 2010 年起，我陆续将剩下的几篇写完。

出书比写博客要求高很多。一本好书需要结构系统而文字严谨，为了达到出书的要求，我几乎重写了所有的内容。因此，这本书虽然在每章的标题和主题上与原来的博客相同，但是内容和文字都是新的。希望广大读者，无论是过去读过黑板报上连载系列的老朋友，还是第一次读这本书的新朋友，都能有全新的收获。

在系统性方面，为了便于非 IT 读者的阅读，我对每个专题都给出了背景介绍；同时，为了起到给从事相关工作的工程师做参考的目的，在一些专题的最后，我都给出了“延伸阅读”一节。非 IT 读者可以完全跳过这些延伸阅读部分，这样并不会影响阅读其他内容。本书中系统性方面的第二个改进就是调整了章节的位置，以帮助读者阅读。在严谨性方面，我在腾讯工程师王益等人的帮助下，更正了原来博客中的一些错误，并尽可能补充完善很多公式推导的过程。

本书的素材来源于我本人的工作。语言信息处理、互联网技术、数据挖掘和机器学习都是博大精深而又快速发展的领域，我所做的研究工作也只涵盖了其中很小的一部分。对于我没有涉足过的领域，我没有信心也没有资格写。因此，这本书在内容上并没有全面覆盖上述领域，比如对当今数据挖掘领域的算法、互联网上各种推荐系统的数学模型都鲜有提及。对这些内容有兴趣的读者可以查阅相应的书籍文章，也希望今后有这方面的专家能够将自己工作的心得写出来，供大家学习参考。

在《数学之美》成书之际，我要感谢所有那些把我带到数学王国和信息处理领域的人。特别要感谢的是我的父亲，他让我在幼年就对数学和自然科学产生了浓厚的兴趣，并帮助我打好数学基础。接下来要特别感谢的是我在中国和美国的三位导师：王作英教授、库坦普教授和贾里尼克教授，他们三人都是有着很深数学造诣的信息论专家，他们把我领进语音和语言处理的王国，不仅帮助我打下比较深厚的数学功底，而且让我看到了数学的威力和魅力。最后要感谢 Google 的谢尔盖·布林院士、韦恩·罗森先生和彼得·诺威格博士，他们把我招进Google，让我有机会

进入网络搜索领域。尤其是诺威格博士，作为我在Google的直接上级，一直给予我做任何我想做的工作的权力，因此，我才得以精通网络搜索的每一个细节。在Google，我还要感谢我的同事阿米特·辛格博士，他是我在搜索领域的导师。

在这本书的写作过程中，我得到了很多人的帮助和鼓励。Just Pub团队的审稿编辑李琳骁先生对书稿进行了数次精心的审读，感谢他的认真和专业。华中科技大学数学与统计学院的周笠教授也拨冗通读了全书，感谢他的鼓励和细心。腾讯公司的王益博士（原Google工程师）帮我编辑和校对了书中诸多的公式，并完善了书中的一些细节部分。吴丹丹女士在将我的拙作刊登在谷歌黑板报前，对我的博客进行了润色，尤其是增加了不少画龙点睛之笔。在我将博客编辑成书时，Google的叶艳女士通读了所有的章节，并给出了修改意见，以保证全书对于非专业读者的可读性。

清华大学的李星教授、李开复博士、本书的出版人周筠女士和原Google中国的总监王怀南先生一直是这个博客系列的热心推荐人，并且一直鼓励我将它编纂成书。我以前在清华大学的同事郭进博士是自然语言处理专家，经常和我讨论中文处理的问题，并且给了我很多启发。在此，我向他们表示诚挚的感谢。

同时我要感谢我的家人给我的帮助，特别是我的两个女儿吴梦华和吴梦馨，她俩绘制了全书的许多插图。

最后要感谢所有热心的博客读者以及在互联网上传播这个博客的媒体、网站和个人。希望这本书能够帮助读者从广度和深度上了解信息科学。

吴军

2012年4月于深圳

序言 1

《数学之美》是一本非常值得读的书。这本书展现了吴军博士在他多年的科研经历中对科学问题的深入思考。

我于1991年从美国回到清华大学电子工程系工作，与吴军博士是同事，对他在汉语语音识别方面的深入研究印象非常深刻。后来他到美国工作，出版了一本介绍硅谷的书《浪潮之巅》，使我对他的写作激情和水平有了新的认识。

这些年来我在清华大学教书，一直思考着如何让学生能真正欣赏和热爱科学研究，这将有助于他们深入理解自己所从事的研究的价值，进而能逐渐成长为所在领域的大师和领军人物。在这一过程中，恰好发现了吴军博士在谷歌中国的官方博客——谷歌黑板报上连载的“数学之美”系列文章，我非常欣赏这些文章。因此，在很多场合都建议学生跟踪阅读这个系列的博客文章。今天本书出版，与原先的博客文章相比，其内容的系统性和深度又上升到了一个新的境界。

我读《数学之美》有下面几点体会，与大家分享。

1. 追根溯源

《数学之美》用了大量篇幅介绍各个领域的典故，读来令人兴趣盎然。典故里最核心的是相关历史事件中的人物。我们必须要问：提出巧妙数学思想的人是谁？为什么是"他 / 她"提出了这个思想？其思维方法有何特点？成为一个领域的大师有其偶然性，但更有其必然性。其必然性就是大师们的思维方法。

2. 体会方法

从事科学研究，最重要的是掌握思维方法。在这里，我举两个例子。

牛顿是伟大的物理学家和数学家，他在《自然哲学的数学原理》中叙述了四条法则。其中有"法则 1：除那些真实而已足够说明其现象者外，不必去寻找自然界事物的其他原因"。这条法则后来被人们称作"简单性原则"。正如爱因斯坦所说："从希腊哲学到现代物理学的整个科学史中，不断有人力图把表面上极为复杂的自然现象归结为几个简单的基本概念和关系。这就是整个自然哲学的基本原理。"这个原理也贯穿了《数学之美》本身。

WWW 的发明人蒂姆·伯纳斯·李谈到设计原理时说过："简单性和模块化是软件工程的基石；分布式和容错性是互联网的生命。"虽然在软件工程和互联网领域的从业人员数量极其庞大，但能够真正体会到这些核心思想的人能有多少呢？

我给学生出过这样的考题：把过去十年来重要 IT 杂志的封面上重点推荐的技术专题找来看看，瞧一瞧哪些技术成功了，哪些技术是昙花一现，分析一下原因？其答案很有意思："有正确设计思想方法的技术"未必能够成功，因为还有非技术的因素；但"没有正确设计思想方法的技术"一定失败，无一例外。因此，我也建议本书的读者结合阅读，体会凝练创造《数学之美》的方法论。

3. 超越欣赏

数学既是对自然界事实的总结和归纳，如英国的哲学家培根所说“一切多依赖于我们把眼睛紧盯在自然界的事实之上”；又是抽象思考的结果，如法国哲学家笛卡尔所说“我思故我在”。这两个方法成就了目前绚丽多彩、魅力非凡的数学，非常值得欣赏。《数学之美》把数学在 IT 领域，特别是语音识别和搜索引擎方面的美丽之处予以了精彩表达。但在这里我想说的是：欣赏美不是终极目的，更值得追求的是创造美的境界。希望本书的读者，特别是年轻读者能够欣赏数学在 IT 技术中的美，学习大师们的思想方法，使自己成为大师，创造新的数学之美。

李星

2012 年 4 月于北京

序言 2

去年我曾经给吴军的《浪潮之巅》写序，今年很高兴得知他的《数学之美》也即将出版了！

和《浪潮之巅》一样，《数学之美》也是当年作为 Google 资深研究员的吴军在谷歌黑板报上应邀撰写的一系列文章。说实在的，刚开始，黑板报的版主还有点担心这个系列会不会让读者觉得太理论而感到枯燥，但很快这个顾虑就被打消了。《数学之美》用生动形象的语言，结合数学发展的历史和实际的案例，谈古论今，系统地阐述了与现代科技领域相关的重要的数学理论的起源、发展及其作用，深入浅出，受到广大读者尤其是科技界人士的喜爱。

之前就曾说过，在我认识的顶尖研究员和工程师里，吴军是极少数具有强大叙事能力和对科技、信息领域的发展变化有很深的纵向洞察力，并能有效归纳总结的人之一。在《数学之美》里，吴军再次展示了这一特点。与《浪潮之巅》不同的是，这次吴军集中阐述了他对数学和信息处理这些专业学科的理解，尤其是他在语音识别、自然语言处理和信息搜索领域多年来的积累 。 从数字和信息的由来，到搜索引擎对信息进行处理背后的数学原理，到与搜索相关的众多领域后面

的奇妙的数学应用，吴军都娓娓道来。他把数学后面的本质思维写得透彻、生动。不得不说，他的文字，引人入胜，也确实让我们体会到数学的美。在他的笔下，数学不是我们一般联想到的枯燥深奥的符号，而是实实在在源于生活的有趣的现象和延伸。数学，其实无处不在，而且有一种让人惊叹的韵律和美！

伽利略曾经说过，“数学是上帝描写自然的语言”；爱因斯坦也曾说过，“纯数学使我们能够发现概念和联系这些概念的规律，这些概念和规律给了我们理解自然现象的钥匙。”我多年来一直也对信息处理、语音识别领域有着一定的研究，深深体会到数学在所有科学领域起到的基础和根本的作用。“哪里有数，哪里就有美”。在这里，我把《数学之美》真诚推荐给每一位对自然、科学、生活有兴趣有热情的朋友，不管你是搞理科还是搞文科的，读一读数学的东西，会让你受益良多，同时能感受到宇宙和世界的美好与奇妙。

吴军把之前谷歌黑板报上的“数学之美”系列文章编辑成现在的这本书，花费了大量的心血和时间。他本着十分严谨的态度，在繁忙的工作之余，补充了之前的系列，并几乎重写了所有的文章，既照顾了普通读者的兴趣，又兼顾了专业读者对深度的要求，很让人钦佩。

有时我在想，现在的社会多了一点压力和浮躁，少了一点踏实和对自然科学本质的好奇求知。吴军的这本《数学之美》真的非常好。非常希望吴军今后能写出更多这样深入浅出的好书，它们会是给这个社会和年轻人最好的礼物。

李开复

2012年4月于北京

前言

数学一词在西方源于古希腊语 μάθημα，意思是通过学习获得的知识的意思，因此早期的数学涵盖的范围比我们今天讲的数学要广得多，和人类的生活也更接近些。在古代最重要的知识，除了对世界的认识和了解，就是人之间的互通和交流了，我们把它称为广义上的通信。本书的内容也将从这里开始。

早期的数学远不如今天神秘，它是非常真实的。但是和任何事物一样，数学也在不断地演化，而这个发展过程使得数学变得高深起来。数学演化的过程实际上是将我们生活中遇到的具体物质以及他们运动的规律不断抽象化的过程。经过几千年的抽象化，大家头脑里能想象的数学只剩下数字、符号、公式和定理了。这些东西和我们的生活似乎渐渐疏远了，甚至在表面上毫不相关了。今天，除了初等数学，大家一般对数学尤其是纯粹数学（Pure Mathematics）的用途甚至产生了怀疑。很多大学生毕业后，在大学所学的数学可能一辈子都没有机会应用，几年后就忘得差不多了。因此，很多人也产生了为什么要学习数学的疑问。更加不幸的是，数学专业的毕业生就连就业也颇为困难，在中国和美国都是如此。在很多人眼里，数学家都是陈景润那样带着厚厚的眼镜、行为木讷的人。因此，无论是这些抽象的数字、符号、

公式和定理，还是研究他们的数学家和美也似乎没有联系。

事实上数学的用途远不止人们的想象，甚至可以说在我们生活中是无所不在。且不说那些和我们生活相对联系较少的领域，比如原子能和航天，那里需要用到大量的数学知识。就说我们天天用的产品和技术，背后都有支持它们的数学基础。作为一名工作了 20 多年的科学工作者，我在工作中经常惊叹于数学语言应用于解决实际问题上时的魔力。我也希望把这种神奇讲解给大家听。

从工业社会起，通信占据了人们生活的大量时间。当人类进入电的时代后，通信的扩展不仅拉近了人与人的距离，而且是带动世界经济增长的火车头。今天通信和它相关的产业可能占到我们世界 GDP 很大的一部分。今天城市里的人花时间最多无非是在电视机前，互联网上，电话上（不论是固定电话还是手机），这些都是这样或者那样的通信。甚至原本必须人到现场的很多活动比如购物，也被建立在现代通信基础之上的电子商务逐渐取代。而现代通信，追溯到 100 多年前的莫尔斯电报码和贝尔的电话，再回到今天的电视，手机和互联网，都遵循信息论的规律，而整个信息论的基础就是数学。如果往更远看，我们自然语言和文字的起源背后都受着数学规律的支配。

“信”字作为“通信”一词的 50%，表明了信息处理存储、传输、处理和理解的重要性。我们今天每个人都使用的搜索，以及我们都觉得很神奇的语音识别、机器翻译和自然语言处理也被包括在其中。也许大家不相信，数学是解决这些问题的最好工具。它不仅能够非常清晰地用一些通用的模型来描述这些领域的看似不同的实际问题，而且能给出非常漂亮的解决办法。每当人们应用数学工具解决一个个和信息处理有关的问题时，总会感叹数学之美。虽然人类的语言有成百上千种，但处理它们的数学模型却是相同的或者相似的，这种一致性也是数学之美的表现。在这本书中，我们将介绍一些数学工具，看看我们是如何利用这些工具来处理信息，开发我们每天生活中都使用的产品。

数学常常给人一种深奥和复杂的感觉，但是它的本质常常是很简单而直接的。英国哲学家弗朗西斯·培根在论美德时讲“美德就如同华贵的宝石，在朴素的衬托下最显华丽。”（Virtue is like a rich stone, best plain set.），数学的妙处也恰恰在于一个好的方法，常常是最简单明了的方法。因此，我会将简单即是美的思想贯穿全书。

最后，要说明一下本书为什么花了相当的篇幅介绍很多我所熟知的自然语言处理和通信的世界级专家。他们来自世界不同的国家，属于不同的民族，但是他们都有一个共同的特点就是数学非常好，同时解决了很多实际问题。通过介绍他们日常的工作和生活，希望读者对真正的世界级学者有更多的了解。了解他们凡人的一面，了解他们成功的原因，了解真正懂得数学之美的人的美好人生。

吴军

2012年4月于深圳

目录

第1章　文字和语言 vs 数字和信息

数字、文字和自然语言一样，都是信息的载体，它们之间原本有着天然的联系。语言和数学的产生都是为了同一个目的 —— 记录和传播信息。但是把数学和信息系统自觉地联系起来是半个多世纪前香农博士发明信息论以后的事。在此之前，数学的发展更多地和人类对自然的认识以及生产活动联系在一起，包括天文学、几何和工程学、经济学、力学、物理学甚至生物学等，而数学和语言学几乎是没有交集的。我们见到很多数学家同时是物理学家或者天文学家，但是过去很少有数学家同时是语言学家。

本书几乎全部的章节讲的都是近半个多世纪的事情，但是在这一章里，我们将先通过时间隧道回到远古，回到语言、文字和数字产生的年代。

1　信息

我们的祖先“现代人”（人类学上的说法）在长成我们今天的模样以前，就开始使用和传播信息了。正如动物园里的动物们经常发出它们喜欢的怪叫一样，早期的人类也喜欢发出含糊的声音。虽然最初可能只是喜欢这样发声，渐渐地人类开始用这种声音来传播信息，比如用某种特定的声音表示“那里有只熊”，提醒同伴小心。同伴可能“呀呀”地回应两声，

表示知道了，或者发出另一串含糊不清的声音，表示“我们用石头打它”。



图 1.1　人类最早利用声音的通信

这里面信息的产生、传播、接收和反馈，与今天最先进的通信在原理上没有任何差别。关于信息传播的模型，在以后的章节中还会详细介绍。

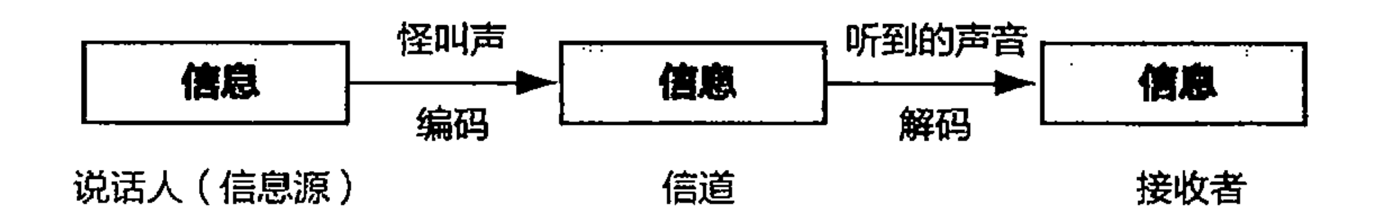


图 1.2　原始人通信的方式和今天的通信模型没有什么不同

早期人类了解和需要传播的信息是很少的，因此他们并不需要语言和数字。但是随着人类的进步和文明化的进展，需要表达的信息也越来越多，不再是几种不同的声音就能完全覆盖，语言就此产生。人们生活的经验，作为一种特定的信息，其实是那个时代最宝贵的财富，通过口述的语言传给了后代。同时，由于人类开始拥有一些食物和物件，便有了多和少的概念。很遗憾，那时的人类还不会数数，因为他们不需要。

2　文字和数字

我们的祖先迅速学习新鲜事物，语言也越来越丰富，越来越抽象。语言描述的共同因素，比如物体、数量、动作便抽象出来，形成了今天的词汇。

当语言和词汇多到一定程度的时候，人类仅靠大脑已经记不住所有词汇了。这就如同今天没有人能够记住人类所有的知识一样。高效记录信息的需求就产生了，这便是文字的起源。

这些文字（包括数字）出现的年代，今天是可以考证的。很多读者问我为什么在《浪潮之巅》一书中讲的公司大多在美国，因为近百年的技术革命大多发生在那里。不过，要提到 5 000 甚至 10 000 年前的信息革命时，我们必须回到我们祖先走出的大陆 —— 非洲，那里是人类文明的摇篮。

在中国（迄今发现的）最早的甲骨文[1]出现前的几千年，尼罗河流域就有了高度的文明。古埃及人不仅是优秀的农夫和建筑师，他们还发明了最早的保存信息的方式 —— 用图形表示事物，这就是最早的象形文字（Hieroglyphic）。图 1.3 是古埃及的“亚尼的死者之书”（Book of The Death），收藏于大英博物馆，这是一轴绘在纸莎草纸上长达 20 多米的长卷，有 60 幅绘画故事和象形文字的说明。这件 3 300–3 400 年前的文物，完整地记载了当时的文明[2]。

1 即大辛庄甲骨文。它的年代应不晚于殷墟文化三期，距今约 3 200 年。



图 1.3　亚尼的死者之书

2 死者之书是随葬品，放在棺中，可以看作是古埃及死者带到另一个世界的介绍信和今后生活的描述。上面有大量的象形文字，内容大致是说，死者被带到冥神面前，首先，他向冥神讲述他一辈子没有做任何坏事，然后来到诸神面前被裁判，最后搭乘太阳船开始新的生活。任何人看到它后都会感到震撼。它的制作之精美，保存之完好完全超出我们的想象。

在早期，象形文字的数量和记录一个文明需要的信息量显然是相关的。最早刻有埃及象形文字的文物的年代大约是公元前 32 世纪，那个时期的象形文字数量大约只有 500 个，但是到了公元前 5-7 世纪（主要是“希腊 - 罗马时代”，Greece-Roman Era），埃及象形文字的数量增加到了 5 000 个左右，这个规模和中国常用的汉字数量相当[3]。然而随着文明的进步，信息量的增加，埃及的象形文字数量便不再随着文明的发展而增加了，因为没有人能够学会和记住这么多的文字。于是，概念的第一次概括和归类就开始了。在中国的象形文字中，“日”本意是太阳，但是它同时又是太阳从升起到落山再到升起的时间周期，也就是我们讲的一天。在古埃及的象形文字中，读音相同的词可能用同一个符号记录。这种概念的聚类，在原理上与今天自然语言处理或者机器学习的聚类有很大的相似性，只是在远古，完成这个过程的时间可能需要上千年；而今天，可能只需几天甚至几小时，视计算机的速度和数量而定。

3 以二级国标汉字库为准。

文字按照意思来聚类，最终会带来一些歧义性，也就是说有时弄不清一个多义字在特定环境下它到底表示其中的哪个含义。而解决这个问题的方法，过去的先生和今天的学者没有什么不同，都是依靠上下文。有了上下文，大多数情况下多义字的去除歧义（Disambiguation）可以做到。当然，总有个别做不到的时候，这就导致了学者们对某段话理解上的不同。中国古代学者对儒家经典的注释和正义，其实都是在按照自己的理解做消除歧义性的工作。今天的情况也类似，对上下文建立的概率模型再好，也有失灵的时候。这些是语言从产生伊始就固有的特点。

有了文字，前人的生活经验和发生的事件便一代代传了下来。只要一个文明不中断，或者这种文字还有人认识，这些信息就会永远地传下去，比如中国的文明便是如此。当然，当一种文字不再有人认识时，破解相应的信息就有点困难了，虽然办法还是有的。

不同的文明，因为地域的原因，历史上相互隔绝，便会有不同的文字。随着文明的融合与冲突，不同文明下的人们需要进行交流，或者说通信，那

么翻译的需求便产生了。翻译这件事之所以能达成，仅仅是因为不同的文字系统在记录信息上的能力是等价的。(这个结论很重要。)进一步讲，文字只是信息的载体，而非信息本身。那么不用文字，而用其他的载体（比如数字）是否可以存储同样意义的信息呢？这个答案是肯定的，这也是我们今天现代通信的基础。当然，不同的文明进行交流时，或许会用不同的文字记载同一件事情。这就有可能为我们破解无人能懂的语言提供一把钥匙。

从公元前 7 世纪起，随着希腊人开始卷入埃及的政权之争[4]，希腊文化开始对埃及产生了影响。尤其是后来希腊人（包括马其顿人）和罗马人先后成了埃及的主人，埃及的语言也逐渐拉丁化。象形文字退出了历史的舞台，不再是人们通信的工具，而只是一种信息的记载，只有庙里的祭司们能认得了。到了公元 4 世纪左右，罗马皇帝迪奥多西一世下令在埃及清除非基督教的宗教，埃及的象形文字从此失传。1 400 多年后，1798 年，拿破仑的远征军来到埃及，随军有上百名学者。一天，一个叫皮埃尔·弗朗索瓦·布沙尔（Pierre-François Bouchard）的中尉在一个叫罗塞塔（Rosetta）的地方发现了一块破碎的古埃及石碑（图 1.4），上面有三种语言：埃及象形文字、埃及的拼音文字和古希腊文。他意识到这块石碑对破解古埃及秘密的重要性，便交给了随行的科学家让·约瑟夫·马塞尔（Jean-Joseph Marcel），后者拓下了石碑上的文字带回法国。虽然 1801 年随着法国在埃及战败，罗塞塔石碑从法国人手里转到了英国人手里[5]，但是马塞尔带回的拓片却在法国和欧洲的学者中传阅，直到 21 年后的 1822 年，法国语言学家商博良（Jean-François Champollion）破解了罗塞塔石碑上的古埃及象形文字。可见文字本身的载体是石头还是纸张并不重要，它所承载的信息才是最重要的。

4
公元前 653 年，希腊商人帮助埃及人抵抗外族入侵。

5
这是今天大英博物馆镇馆之宝之一。



图 1.4 罗塞塔石碑

罗塞塔石碑的破译，让我们了解了整个埃及从公元前 32 世纪（早期王朝时代）至今的历史，这是让历史学界和语言学界最振奋的事情。今天，我们对 5 000 年前埃及的了解远比 1 000 年前的玛雅文明要多得多，这要归功于埃及人通过文字记录了他们生活中最重要的信息。而对于我这个长期从事自然语言处理的学者来讲，这件事有两点指导意义。

1. 信息的冗余是信息安全的保障。罗塞塔石碑上的内容是同一信息重复三次，因此只要有一份内容完好保留下来，原有的信息就不会丢失，这对信道编码有指导意义。（感谢 2 000 多年前古埃及人在罗塞塔石碑上用三种文字记录了托勒密五世登基的诏书。）

2. 语言的数据，我们称之为语料，尤其是双语或者多语的对照语料对翻译至关重要，它是我们从事机器翻译研究的基础。在这个方法上，我们并没有比商博良走得更远。唯一不同的是，我们有了更强大的数学工具和计算机，不需要花费他那么长的时间。

了解了罗塞塔石碑的历史，对于今天很多翻译软件和服务都叫做“罗塞塔”就不奇怪了，这其中包括 Google 的机器翻译和世界上销量最大的 PC 机上的翻译软件。

既然文字是出现在远古“信息爆炸”导致人们的头脑装不下这些信息的时候，那么数字的出现则是在人们的财产多到需要数一数才搞清楚有多少的时候。著名的美籍俄裔物理学家乔治·伽莫夫（George Gamow，1904-1968）在他的科普读物《从一到无穷大》一书中讲了这样一个原始部落中的故事。两个酋长要比一比谁说的数字大，一个酋长想了想，先说了“三”，第二个酋长想了半天，说你赢了。因为在原始部落，物质极其缺乏，超过三的时候很少，他们就称之为“许多”或者叫数不清。因此，在那个时代，不可能有完整的计数系统。

当我们的祖先需要记录的物件超过三时，当他们觉得五和八还是有区别的时候，计数系统就产生了。而数字是计数系统的基础。当然，早期数字并没有书写的形式，而是掰指头，这就是为什么我们今天使用十进制的原因。毫无疑问，如果我们有十二个指头，今天我们用的一定是十二进制。而具有书写形式的数字和象形文字应该诞生于同一时期 —— 几乎所有早期的文明对于数字 1、2、3 的记录方式都是几横（中国）、几竖（罗马）或者几个楔子点（巴比伦，图 1.5），这是典型的象形文字的特征。因此，数字和其他文字一样，在早期只是承载信息的工具，并不具有任何抽象的含义。

图 1.5　古巴比伦的数字

渐渐地，我们的祖先发现十个指头不够用了。虽然最简单的办法就是把十个脚趾头也算上，但是这不能解决根本问题。事实上，我们的祖先没有这样做，当然也许在欧亚非大陆上出现过这么做的部落，但早就被灭绝了。我们的祖先很聪明，他们发明了进位制，也就是我们今天说的逢十进一。这是人类在科学上的一大飞跃，因为我们的祖先懂得对数量开始编码了，不同的数字代表不同的量。几乎所有的文明都采用了十进制，那么是否有文明采用二十进制呢，也就是说他们数完全部的手指和脚趾才开始进位呢？答案是肯定的，这就是玛雅文明。因此，玛雅人的一个世纪，他们称为太阳纪，是四百年。2012 年正好是目前这个太阳纪的最后一年，2013 年将是新的太阳纪的开始。这是我在墨西哥从研究玛雅文化的教授那里得知的。不知道从何时起，2012 年这个太阳纪的最后一年被讹传为世界的最后一年。当然，这是题外话了。

相比十进制，二十进制有很多不便之处。我们中国人过去即使是不识几个字的人，也能背诵九九表。但是，换成二十进制，要背的可就是 19×19 的围棋盘了。事实上，很多过去使用英制[6]的英联邦国家，比如印度、斯里兰卡，乘法表是 12×12 的。即使到了人类文明的中期，即公元前后，除非是学者，几乎没有人能够做到这一点。我想这可能是玛雅文明发展非常缓慢的原因之一，当然更重要的原因是它的文字极为复杂，以至于每个部落没有几个人能认识。

6 十进制、十二进制和十六进制混杂在一起的复杂单位制，除了美国，今天已经没有主要国家使用了。

对于不同位数数字的表示，中国人和罗马人都用明确的单位来表示数字的不同量级，中国人是用个十百千万亿兆[7]。罗马人用 I 代表个，V 代表 5，X 代表 10，L 代表 50，C 代表 100，D 代表 500，M 代表 1000，再往上就没有了。这两种表示法都不自觉地引入了朴素的编码的概念。首先，它们都是用不同的符号代表不同的数字概念；第二，它们分别制定了解码的规则。在中国，解码的规则是乘法。200 万的写法含义是 2×100×10000；而在罗马，解码的规则是加减法 —— 小数字出现在大数字左边为减，右边为加。比如 IV 表示 5－1＝4，VII 表示 5＋2＝7，IIXX 表示 20－2＝18。这个规则不仅复杂，而且对于大的数字很难描述。罗马人要写 100 万的话，恐怕要 MMMM…地不断写下去，写满一整块黑板，直到近代他们才在 M 上用上划线表示几万和几十万。因此，从编码的有效性来讲，中国人的做法比罗马人高明[8]。

7 兆本身又有两个含义：百万和万亿。

8 实际上，罗马人的老师希腊人的计数方式和中国古代颇为相似，不知为什么，罗马人没学会。

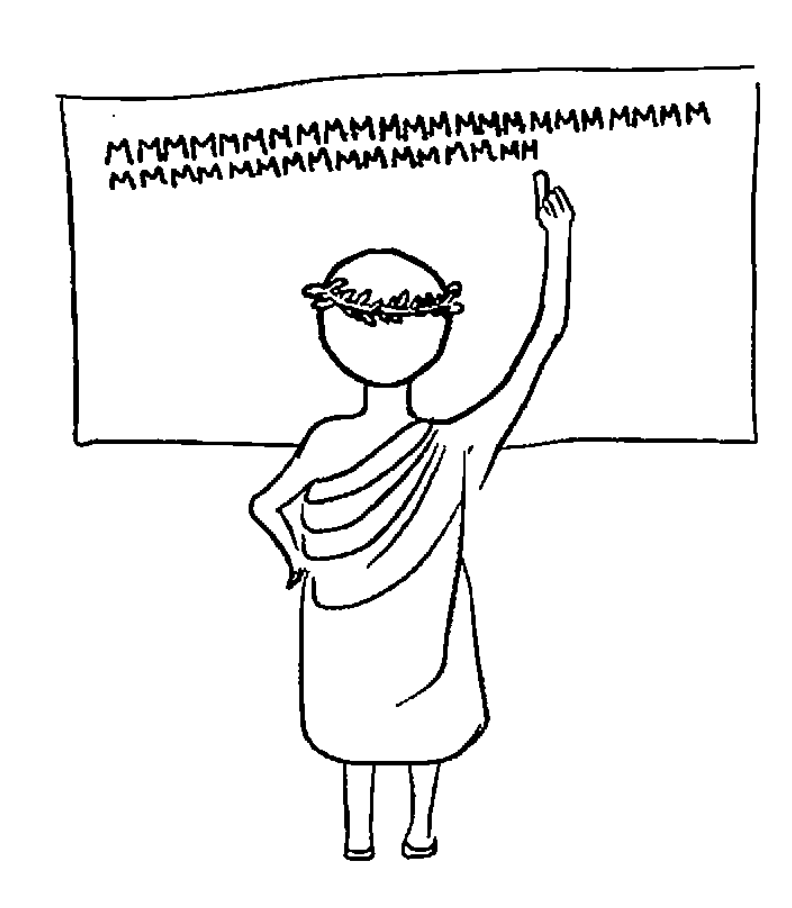


图 1.6　一位罗马学者试图在黑板上写出 100 万

描述数字最有效的是古印度人，他们发明了包括 0 在内的 10 个阿拉伯数字[9]，就是今天全世界通用的数字。这种表示方法比中国和罗马的都抽象，但是使用方便。因此，它们由阿拉伯人传入欧洲后，马上得到普及。只是欧洲人并不知道这些数字的真正发明者是印度人，而把功劳给了“二道贩子”阿拉伯人。阿拉伯数字或者说印度数字的革命性不仅在于它的

9 这个 0 很重要，否则就需要许多描述进制的量词，如个十百千万等。

简洁有效，而且标志着数字和文字的分离。这在客观上让自然语言的研究和数学在几千年里没有重复的轨迹，而且越走越远。

3 文字和语言背后的数学

但是，任何事物的规律性是内在的，并不随它的载体而改变。自然语言的发展在冥冥之中，都受着信息科学规律的引导。

当人类第二个文明的中心古巴比伦在两河流域的美索不达米亚建立的时候，一种新的文字 —— 楔形文字诞生了。考古学家和语言学家最初看到这些刻在泥板和石板[10]上，与埃及象形文字多少有点相像的符号时，以为它们是另一种象形文字。但是很快他们就发现这些文字其实是拼音文字，是我们这个星球上最古老的拼音文字，每个形状不同的楔子实际上是一个不同的字母。如果把中文的笔画作为字母，它其实也是一种拼音文字，不过它是二维的而已。（不过，为了和罗马体系的拼音文字相区别，在这本书中我们会把汉字称为意型文字。）大英博物馆保存了上万块这种泥板和石板，上面都是楔形文字。这些刻满了文字的石板和泥板与亚述浮雕一起，被认为是最有价值的古巴比伦文物。

10 最早的泥板年代在公元前 26 世纪或者更远，迄今有 4 700 年左右的历史。

拼音文字由腓尼基人从以古巴比伦为中心的美索不达米亚带到地中海东岸的叙利亚。腓尼基人是天生的商人，不愿意把大量的时间花在雕刻这些漂亮的楔形字母上，而将它们简化成 22 个字母。这些字母随着腓尼基人的商团经爱琴海诸岛（如克里特岛），然后传给了希腊人的祖先。但是，拼音文字在古希腊得到了充分的发展，和古巴比伦的楔形字母已经不同，古希腊文字母的拼写和读音已经紧密地结合起来了，这种语言相对容易学习。在今后的几个世纪里，伴随着马其顿人以及几个世纪后的罗马人的扩张，这些只需要几十个字母的语言成为了欧亚非大陆语言体系的主体，因此，今天我们把所有西方的拼音文字称为罗马式的语言（Roman Languages）。

从象形文字到拼音文字是一个飞跃，因为人类在描述物体的方式上，从

物体的外表到抽象的概念，同时不自觉地采用了对信息的编码 。不仅如此，我们的祖先对文字的编码还非常合理。在罗马体系的文字中，总体来讲，常用字短，生僻字长。而在意型文字中，也是类似，大多常用字笔画少，而生僻字笔画多。这完全符合信息论中的最短编码原理，虽然我们的祖先并不懂信息论。这种文字设计（其实是一种编码方法）带来的好处是书写起来省时间、省材料。

在蔡伦发明纸张以前，书写文字不是一件容易的事情。就以中文为例，在东汉以前要将文字刻在其他物件比如龟壳、石碑和竹简上。由于刻一个字的时间相当长，因此要惜墨如金。这就使得我们的古文（书面文字）非常简洁，但是非常难懂，而同时期的口语却和今天的白话差别不大，语句较长但是易懂。（岭南客家话基本上保留了古代口语的原貌，写出来和我们清末民初的白话颇为相似。）这种现象非常符合今天信息科学（和工程）的一些基本原理，就是在通信时，如果信道较宽，信息不必压缩就可以直接传递；而如果信道很窄，信息在传递前需要尽可能地压缩，然后在接收端进行解压缩。在古代，两个人讲话说得快是一个宽信道，无需压缩；书写来得慢是一个窄信道，需要压缩。将日常的白话口语写成精简的文言文本身是信道压缩的过程，而将文言文解释清楚是解压缩的过程。这个现象与我们今天宽带互联网和无线 WAP 互联网网页的设计完全一致，前者是经过一个宽带传输，因此页面设计得较大；而后者由于空中频道带宽的限制，传输速度要慢一到两个数量级，因此 WAP 页面都非常小。由此可见，在信息论尚未被发明的几千年前，中国人已经无意识地遵照它的规律行事了。

当司马迁用近 53 万字记载了中国上千年历史的同时，远在中东的犹太人也用类似的篇幅记载了自创世纪以来，主要是摩西以来他们祖先的历史，这就是《圣经》中的《旧约》部分[11]。《圣经》简洁的文风和中国的《史记》颇有相似之处。但是和《史记》这本由唯一作者写成的史书不同，《圣经》的写作持续了很多世纪，后世的人在做补充时，看到的是几百年前甚至上千年前原作的抄本。抄写的错误便在所难免。据说今天也只有牛津大

11 《圣经》的中译本，不同版本字数不同，大约在 90-100 万字之间。其中《旧约》和《新约》大致各占篇幅的一半，约 50 万字。

学保留了一本没有任何错误的古本。虽然做事认真的犹太人要求在抄写《圣经》时，要虔诚并且打起十二分精神，尤其是每写到“上帝”（God和 Lord）这个词时要去洗手祈祷，但是抄写错误还是难以避免。于是犹太人发明了一种类似于我们今天计算机和通信中校验码的方法。他们把每一个希伯来字母对应于一个数字，这样每行文字加起来便得到一个特殊的数字，这个数字便成为了这一行的校验码。同样，对于每一列也是这样处理。当犹太学者抄完一页《圣经》时，他们需要把每一行的文字加起来，看看新的校验码是否和原文的相同，然后对每一页进行同样的处理[12]。如果这一页每一行和每一列的校验码和原文完全相同，说明这一页的抄写无误。如果某行的校验码和原文中的对应不上，则说明这一行至少有一个抄写错误。当然，错误对应列的校验码也一定和原文对不上，这样可以很快找到出错的地方。这背后的原理和我们今天的各种校验是相同的。

12 Williams, Fred. "Meticulous Care in the Transmission of the Bible." Bible Evidences, n.d. Accessed October 11, 2008.

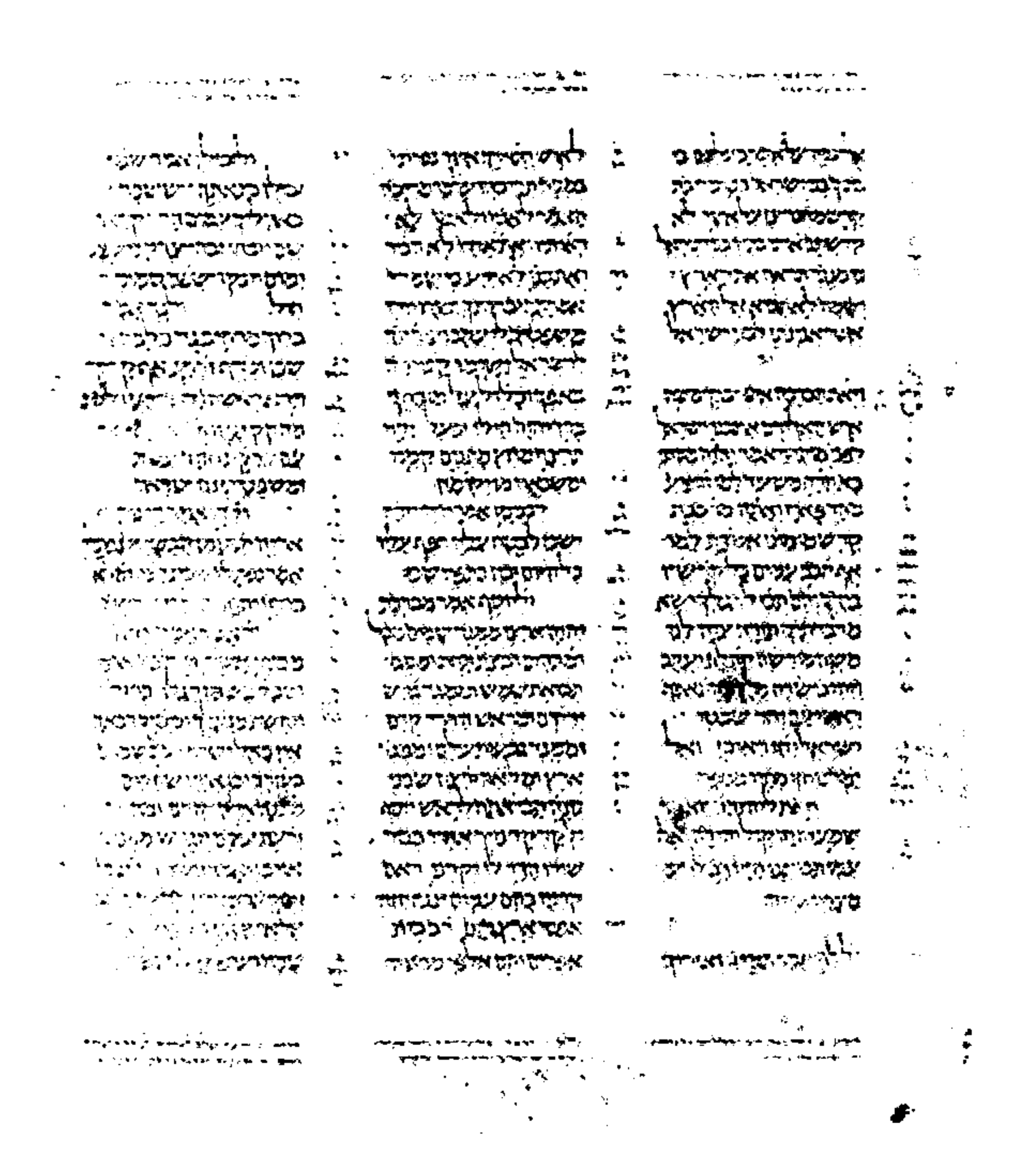


图 1.7 古犹太人抄写圣经，要检查每一行、每一列的校验是否正确

语言从古语发展到现代语言，在表达含义上比以前更准确、更丰富，这里面语法起到了很大的作用。我不是语言史学家，没有去考证最早语法出现的年代，但是大抵可以肯定在古希腊时期就开始成型了[13]。如果说从字母到词的构词法（Morphology）是词的编码规则，那么语法则是语言的编码和解码规则。不过，相比较而言，词可以被认为是有限而且封闭的集合，而语言则是无限和开放的集合。从数学上讲，对于前者可以有完备的编解码规则，而后者则不具备这个特性。因此，任何语言都有语法规则覆盖不到的地方，这些例外或者说不精确性，让我们的语言丰富多彩。虽然正统而教条的语言学家倾向于把这些例外作为“病句”，并且有的人毕其一生的精力来消灭病句，纯化语言，但是事实证明这种工作是徒劳的。莎士比亚的作品在他的时代完全是通俗而大众化的，其中包括大量和古语法相违背的名句，那个时代就开始有人试图完善（其实是篡改）莎士比亚剧。可今天这些语言不但没有消失反而成了经典，而试图完善他的著作的人却早已为大众遗忘。

13 也有人认为在更早的古巴比伦语中就形成了。

这就涉及到一个语言学研究方法的问题：到底是语言对，还是语法对。前者坚持从真实的语句文本（称为语料）出发，而后者坚持从规则出发。经过三四十年的争论，最后实践是检验真理的唯一标准，自然语言处理的成就最终宣布了前者的获胜。这段经过，我们会在第 2 章“自然语言处理 —— 从规则到统计”中介绍。

4　小结

我们在这一章中讲述了文字、数字和语言的历史，目的是帮助读者感受语言和数学天然、内在的联系。这里提到了一些概念和主题，它们是我们在后面的章节中讨论的重点，包括：

- 通信的原理和信息传播的模型
- （信源）编码和最短编码
- 解码的规则，语法

- 聚类
- 校验位
- 双语对照文本，语料库和机器翻译
- 多义性和利用上下文消除歧义性

这些今天自然语言处理学者们研究的问题，我们的祖先在设计语言的时候其实已经遇到了，并且用类似今天的方法解决了，虽然他们的认识大多是自发的，而不是自觉的。他们过去遵循的法则和我们今天探求的研究方法背后有着共同的东西，这就是数学规律。

第 2 章　自然语言处理 —— 从规则到统计

在上一章讲到，语言出现的目的是为了人类之间的通信。字母（或者中文的笔画）、文字和数字实际上是信息编码的不同单位。任何一种语言都是一种编码的方式，而语言的语法规则是编解码的算法。我们把一个要表达的意思，通过某种语言的一句话表达出来，就是用这种语言的编码方式对头脑中的信息做了一次编码，编码的结果就是一串文字。而如果对方懂得这门语言，他或者她就可以用这门语言的解码方法获得说话人要表达的信息。这就是语言的数学本质。虽然传递信息是动物也能做到的，但是利用语言来传递信息是人类的特质。

1946 年[1]，现代电子计算机出现以后，计算机在很多事情上做得比人还好。既然如此，机器是否能够懂得自然语言呢？事实上当计算机一出现，人类就开始琢磨这件事。这里面涉及到两个认知方面的问题：第一，计算机是否能处理自然语言；第二，如果能，那么它处理自然语言的方法是否和人类一样。这本书将回答这两个问题。为了不吊读者的胃口，我在这里先给出简洁版的答案：对这两个问题的回答都是肯定的，Yes！

1 以冯·诺依曼体系的 ENIAC 为准。

1 机器智能

最早提出机器智能设想的是计算机科学之父阿兰·图灵（Alan Turing），1950年他在《思想》（*Mind*）杂志上发表了一篇题为“计算的机器和智能”的论文。在论文中，图灵并没有提出什么研究的方法，而是提出了一种来验证机器是否有智能的方法：让人和机器进行交流（图2.1），如果人无法判断自己交流的对象是人还是机器时，就说明这个机器有智能了。这种方法被后人称为图灵测试（Turing Test）。图灵其实是留下了一个问题，而非答案，但是一般认为自然语言的机器处理（现在称作自然语言处理）的历史可以追溯到那个时候，至今已经60多年了。

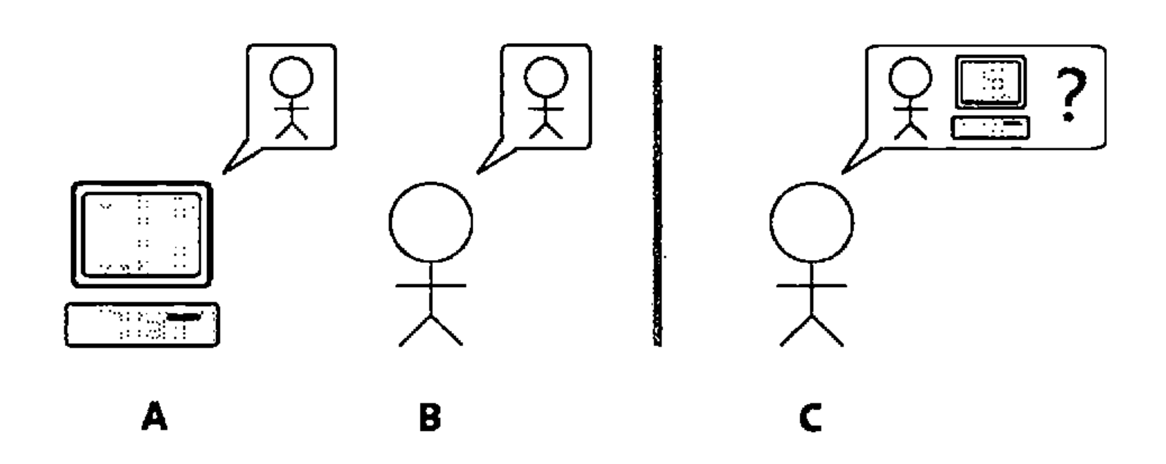


图2.1 搞不清后面是人还是机器

自然语言处理60多年的发展过程，基本上可以分成两个阶段。早期的20多年，即从20世纪50年代到70年代，是科学家们走弯路的阶段。全世界的科学家对计算机处理自然语言的认识都被自己局限在人类学习语言的方式上，也就是说，用电脑模拟人脑，这20多年的成果近乎为零。直到20世纪70年代，一些自然语言处理的先驱重新认识这个问题，找到了基于数学模型和统计的方法，自然语言处理进入第二个阶段。30多年来，这个领域取得了实质性的突破，并且让自然语言处理在很多产品中得以广泛应用。虽然早期自然语言处理的工作对今天没有任何指导意义，但是了解几代科学家的认识过程，对我们了解自然语言处理的方法很有好处，同时也让我们避免走前人的弯路。

让我们回到 1956 年的夏天。28 岁的约翰·麦卡锡（John McCarthy），以及同年龄的马文·明斯基（Marvin Minsky），37 岁的罗切斯特（Nathaniel Rochester）和 40 岁的香农，他们 4 人提议在麦卡锡工作的达特茅斯学院[2]开了一个被他们称为“达特茅斯夏季人工智能研究会议”的头脑风暴式的研讨会。参加会议的还有 6 位年轻的科学家，包括 40 岁的赫伯特·西蒙（Herbert Simon）和 28 岁的艾伦·纽维尔（Allen Newell）。在这次研讨会上，这 10 个人讨论当时计算机科学尚未解决的问题，包括人工智能、自然语言处理和神经网络等。人工智能这个提法便是在这次会议上提出的。这 10 个人，除了香农，当时大多没有什么名气。但是没关系，这些年轻人默默无闻的时间不会太久，后来这些人都非常了不起，其中出了 4 位图灵奖获得者（麦卡锡、明斯基、西蒙和纽维尔）。当然香农不必得什么图灵奖，作为信息论的发明人，他在科学史上的地位和图灵是相当的，而且通信领域的最高奖就是以他的名字命名的。

2 美国常青藤 8 所大学之一，也是美国最好的本科教育学校之一。

达特茅斯会议的意义超过 10 个图灵奖。这 10 位后来被证明是 20 世纪 IT 领域最优秀的科学家，开创了很多今天依然活跃的研究领域，而这些研究领域的成功使我们的生活变得十分美好。遗憾的是，受历史的局限，这 10 个世界上最聪明的头脑一个月的火花碰撞，并没有产生什么了不起的思想，他们对自然语言处理理解的总和，甚至不如今天一位世界一流大学的博士毕业生。这是因为在当时，全世界对自然语言处理的研究都陷入了一个误区。

那时候学术界对人工智能和自然语言理解的普遍认识是这样的：要让机器完成翻译或者语音识别这样只有人类才能做的事情，就必须先让计算机理解自然语言，而做到这一点就必须让计算机有类似我们人类这样的智能。（今天几乎所有的科学家都不坚持这一点，而很多的门外汉还误以为计算机是靠类似我们人类的这种智能解决了上述问题。）为什么会有这样的认识？因为人类就是这么做的，道理就这么简单。对于人类来讲，一个能把英语翻译成汉语的人，一定是能非常好地理解这两种语言的。这就是直觉的作用。在人工智能领域，包括自然语言处理领域，后来把这

样的方法论称作“鸟飞派”，也就是看看鸟是怎样飞的，就能模仿鸟造出飞机，而不需要了解空气动力学。事实上我们知道，怀特兄弟发明飞机靠的是空气动力学而不是仿生学。在这里，我们不要笑话我们前辈来自于直觉的天真想法，这是人类认识的普遍规律。今天，机器翻译和语音识别已经做得不错，并且有上亿人使用过，但是大部分这个领域之外的人依然错误地以为这两个应用是靠计算机理解了自然语言而完成的。事实上，它们全都靠得是数学，更准确地说是靠统计。

在20世纪60年代，摆在科学家面前的问题是怎样才能理解自然语言。当时普遍的认识是首先要做好两件事，即分析语句和获取语义。这实际上又是惯性思维的结果——它受到传统语言学研究的影响。从中世纪以来，语法一直是欧洲大学教授的主要课程之一。到16世纪，伴随着《圣经》被翻译介绍到欧洲以外的国家，这些国家的语言语法逐步得到完善。到18、19世纪，西方的语言学家们已经对各种自然语言进行了非常形式化的总结，这方面的论文非常多，形成了十分完备的体系。学习西方语言，都要学习它们的语法规则（Grammar Rules）、词性（Part of Speech）和构词法（Morphologic）等。当然，应该承认这些规则是我们人类学习语言（尤其是外语）的好工具。而恰恰这些语法规则又很容易用计算机的算法描述，这就更坚定了大家对基于规则的自然语言处理的信心。

对于语义的研究和分析，相比较而言要不系统得多。语义也比语法更难在计算机中表达出来，因此直到20世纪70年代，这方面的工作仍然乏善可陈。值得一提的是，中国古代语言学的研究主要集中在语义而非语法上。很多古老的专著，比如《说文解字》等都是语义学研究的成果。由于语义对于我们理解自然语言是不可或缺的，因此各国政府在把很大比例的研究经费提供给“句法分析”相关研究的同时，也把一部分钱给了语义分析和知识表示等课题。现在把当时科学家头脑里的自然语言处理从研究到应用的依赖关系用图2.2来描述。

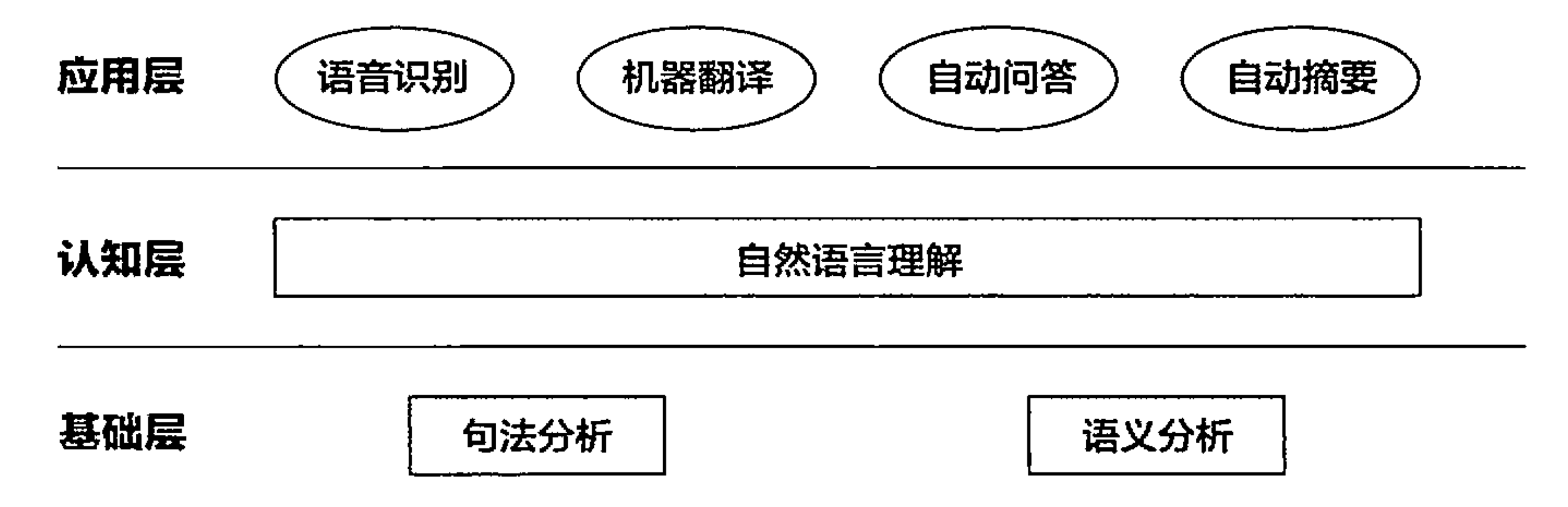


图 2.2 早期对自然语言处理的理解

让我们集中看看句法分析。先看下面一个简单的句子

徐志摩喜欢林徽因。

这个句子可以分为主语、动词短语（即谓语）和句号三部分[3]，然后可以对每个部分作进一步分析，得到如下的语法分析树（Parse Tree）：

3 由于不同的人对同一个句子的句法分析经常不一致，因此，本书凡涉及到句法分析和词性标注等例子，一律以宾夕法尼亚大学语言数据库（LDC）的标准为准。为了方便读者阅读，我把 LDC 的各种表示按实际意义从英语翻译成汉语。

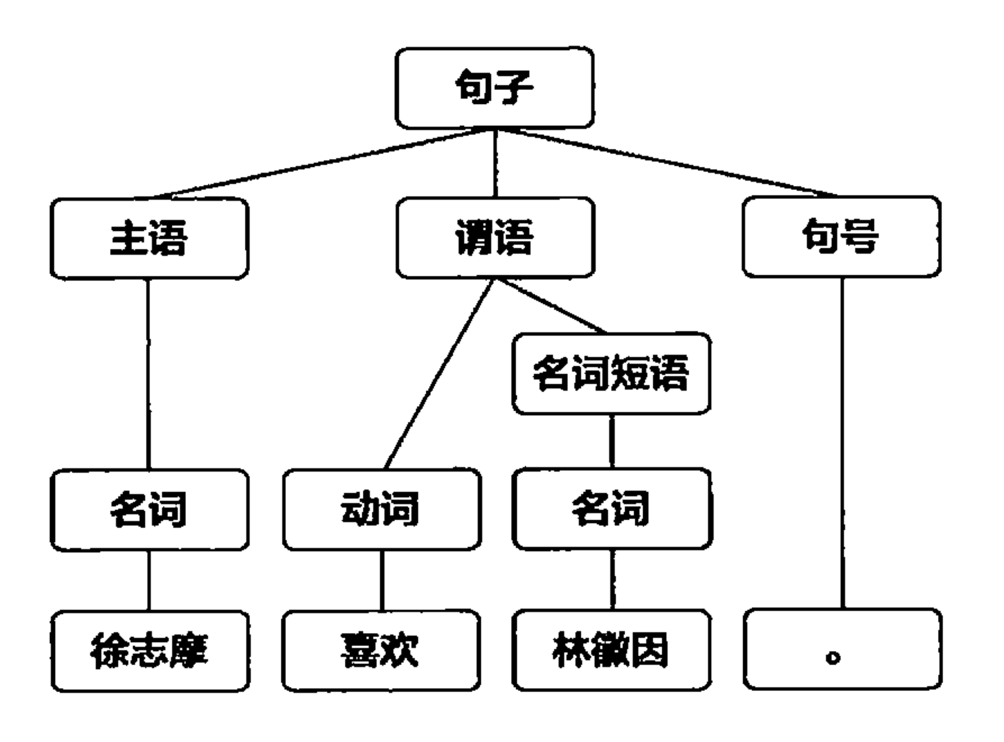


图 2.3 句子的语法分析树

分析它采用的文法规则通常被计算机科学家和语言学家称为重写规则（Rewrite Rules），具体到上面的句子，重写规则包括：

句子→主语谓语句号

主语→名词

谓语→动词 名词短语

名词短语→名词

名词→徐志摩

动词→喜欢

名词→林徽因

句号→。

20 世纪 80 年代以前，自然语言处理工作中的文法规则都是人写的，这和后来采用机器总结的做法大不相同。直到 2000 年后，很多公司，比如著名的机器翻译公司 SysTran，还是靠人来总结文法规则。

20 世纪 60 年代，基于乔姆斯基形式语言的编译器技术得到了很大的发展，计算机高级程序语言都可以概括成上下文无关的文法，这是一个在算法上可以在多项式时间内解决的问题（Polynomial Problem，详见附录）。高级程序语言的规则和上述自然语言的规则从形式上看很相似。因此，很容易想到用类似的方法分析自然语言。那时，科学家们设计了一些非常简单的自然语句的语法分析器（Parser），可以分析词汇表为百十来个词、同时长度为个位数的简单语句（不能有很复杂的从句）。

科学家们原本以为随着对自然语言语法概括得越来越全面，同时计算机计算能力的提高，这种方法可以逐步解决自然语言理解的问题。但是这种想法很快遇到了麻烦。我们从上图可以看出，句法分析实际上是一件很罗嗦的事：一个短短的句子居然分析出这么一个复杂的二维树结构，而且居然需要八条文法规则，即使刨去词性标注的后四条依然还有四条。当然让计算机处理上述分析还是不难的，但要处理下面《华尔街日报》[4] 的一个真实句子就不是那么容易办到了：

> 美联储主席本·伯南克昨天告诉媒体 7 000 亿美元的救助资金将借给上百家银行、保险公司和汽车公司。[5]

虽然这个句子依然符合“句子→主语谓语句号”这条文法规则：

4 在自然语言处理学术领域，各国为了实验结果可以对比，一般采用《华尔街日报》的语料。

5 原句子是这样的 “The Fed Chairman Ben Benanke told media yesterday that $700B bailout funds would be lended to hundreds of banks, insurance companies and automakers.”

主语【美联储主席本·伯南克】|| 动词短语【昨天告诉媒体 7 000 亿美元的救助资金将借给上百家银行、保险公司和汽车公司】|| 句号【。】

然后，接下来可以进行进一步的划分，比如主语“美联储主席本·伯南克”分解成两个名词短语“美联储主席”和“本·伯南克”，当然，前者修饰后者。对于动词短语也可以做同样的分析。这样，任何一个线性的语句，可以被分析成这样一棵二维的语法分析树（Parse Tree）。我们没有将完整的分析树画出来，是因为在这本书一页纸上，无法画出整个语法分析树 —— 这棵树非常大，非常复杂。应该讲，单纯基于文法规则的分析器是处理不了上面这样复杂的语句的。

这里面至少有两个越不过去的坎儿。首先，要想通过文法规则覆盖哪怕 20% 的真实语句，文法规则的数量（不包括词性标注的规则）至少是几万条。语言学家几乎已经是来不及写了，而且这些文法规则写到后来甚至会出现矛盾，为了解决这些矛盾，还要说明各个规则特定的使用环境。如果想要覆盖 50% 以上的语句，文法规则的数量最后会多到每增加一个新句子，就要加入一些新的文法。这种现象不仅出现在计算机处理语言上，而且出现在人类学习和自己母语不同语系的外语时。今天 30 岁以上的人都应该会有这种体会：无论在中学和大学英语考试成绩多么好，也未必能考好 GRE，更谈不上看懂英文的电影。原因就是我们即使学了 10 年的英语语法，也不能涵盖全部的英语。

其次，即使能够写出涵盖所有自然语言现象的语法规则集合，用计算机解析它也是相当困难的。描述自然语言的文法和计算机高级程序语言的文法不同。自然语言在演变过程中，产生了词义和上下文相关的特性。因此，它的文法是比较复杂的上下文有关文法（Context Dependent Grammar），而程序语言是我们人为设计的，为了便于计算机解码的上下文无关文法（Context Independent Grammar），相比自然语言而言简单得多。理解两者的计算量不可同日而语。

在计算机科学中，图灵奖得主高德纳（Donald Knuth）提出了用计算复

杂度（Computational Complexity，请见附录）来衡量算法的耗时。对于上下文无关文法，算法的复杂度基本上是语句长度的二次方，而对于上下文有关文法，计算复杂度基本上是语句长度的六次方。也就是说，长度同为 10 的程序语言的语句和自然语言的语句，计算机对它们进行语法分析的计算量，后者是前者的一万倍。而且随着句子长度的增长，二者计算时间的差异会以非常快的速度扩大。即使今天，有了很快的计算机（英特尔 i5 双核处理器），分析上面这个二三十个词的句子也需要几分钟的时间。因此，在 20 世纪 70 年代，即使是制造大型机的 IBM 公司，也不可能采用规则的方法分析一些真实的语句。

2 从规则到统计

在 20 世纪 70 年代，基于规则的句法分析很快就走到了尽头。而对于语义的处理则遇到了更大的麻烦。首先，自然语言中词的多义性很难用规则来描述，而是严重依赖于上下文，甚至是“世界的知识（World Knowledge）”或者常识。1966 年，著名人工智能专家明斯基（上面提到的达特茅斯会议的发起者之一）举了一个简单的反例，说明计算机处理语言的难处，“The pen is in the box.”和“The box is in the pen.”中两个 pen 的区别。第一句话很好理解，学过半年英语的学生都懂。但是第二句话则会让外国人很困惑，为什么盒子可以装到钢笔里？其实，第二句话对于英语是母语的人来讲很简单，因为这里 pen 是围栏的意思。整句话翻译成中文就是“盒子在围栏里”。这里面 pen 是指钢笔还是围栏，通过上下文已经不能解决，需要常识，具体来说就是“钢笔可以放到盒子里，但是盒子比钢笔大，所以不能放到钢笔里”。这是一个很简单的例子，但是非常明白地说明了当时自然语言处理研究方法上存在的问题。1966 年的明斯基已经不是十年前那个默默无名的年轻人了，而是当时世界上数一数二的人工智能专家。他的意见对美国政府的科技决策部门产生了重大影响，自然科学基金会等部门对传统的自然语言处理研究非常失望，以至于在较长时间里对这方面的研究资助大大减少。可以说，

利用计算机处理自然语言的努力直到 20 世纪 70 年代初是相当失败的。

1970 年以后统计语言学的出现使得自然语言处理重获新生，并取得了今天的非凡成就。推动这个技术路线转变的关键人物是弗里德里克·贾里尼克（Frederick Jelinek）和他领导的 IBM 华生实验室（T. J. Watson）。最初，他们也没有想解决整个自然语言处理的各种问题，而只是希望解决语音识别的问题。采用基于统计的方法，IBM 将当时的语音识别率从 70% 提升到 90%，同时语音识别的规模从几百单词上升到几万单词，这样就使得语音识别有可能从实验室走向实际应用。关于这段历史，后面在贾里尼克的故事里会介绍。

IBM 华生实验室的方法和成就对自然语言处理界的震动是巨大的。后来成为 IBM 和 Google 主管研究的副总裁阿尔弗雷德·斯伯格特（Alfred Spector）博士，当时还是卡内基 - 梅隆大学教授，他 2008 年到 Google 上任后第一次和我单独面谈时，聊起当年这个转变的过程。他说，当年卡内基 - 梅隆大学已经在传统的人工智能领域走得非常远了，大家遇到了很多跨不过去的障碍。后来他去 IBM 华生实验室参观，看到那里依靠统计的方法取得的巨大成绩，连他一个做系统出身的教授也能感受到今后在这个领域研究方法一定要变化。李开复修过他的课，算是他的学生，也是卡内基 - 梅隆大学最早从传统自然语言处理方法转到基于统计方法的人。李开复和洪小文出色的工作，帮助他们的论文导师拉杰·雷迪（Raj Reddy）获得了图灵奖。

作为世界上两个顶尖公司的研究部门第一把手的斯伯格特，是一位对未来研究方向判断力非常敏锐的人，当时看出这一点并不奇怪。但是，并非所有的研究者都认可这个方向。基于规则的自然语言处理和基于统计的自然语言处理的争执后来还持续了 15 年左右，直到 20 世纪 90 年代初。这期间，两路人马各自组织和召开自己的会议。如果在共同的会议上，则在各自的分会场开小会。到 90 年代以后，坚持前一种方法的研究人员越来越少，参会人数自然也越来越少；而后者却越来越多。这样，自然语言

处理从规则到统计的过渡就完成了。15 年，对于一个学者来讲是一段非常长的时间，如果哪个人从做博士开始就选错了方向并且坚持错误，到 15 年后发现时，基本上这一辈子可能就一事无成了。

那么为什么这场争议持续了 15 年呢？首先，一种新的研究方法的成熟需要很多年。上个世纪 70 年代，基于统计的方法的核心模型是通信系统加隐含马尔可夫模型。这一点在后面章节会详细介绍。这个系统的输入和输出都是一维的符号序列，而且保持原有的次序。最早获得成功的语音识别是如此，接下来第二个获得成功的词性分析也是如此。而在句法分析中，输入的是一维的句子，而输出的是二维的分析树；在机器翻译中，虽然输出的依然是一维的句子（另一种语言的），但是次序会有很大的变化，所以上述的方法就不太管用了。1988 年，IBM 的彼得·布朗（Peter Brown）等人提出了基于统计的机器翻译方法[6]，框架是对的，但是效果很差，因为当时既没有足够的统计数据，也没有足够强大的模型来解决不同语言语序颠倒的问题。这样在上个世纪整个 80 年代，除了布朗等人写了这篇论文，没有类似的机器翻译的工作有效展开，而布朗等人也去文艺复兴技术公司[7]发财了。句法分析的问题就更加复杂，因为一个语法成分对另一个语法成分的修饰关系可以不是顺序的，而是中间间隔了很多短语的。只有出现了基于有向图的统计模型才能很好地解决复杂的句法分析。在很长一段时间里，传统方法的捍卫者攻击对方的武器就是，基于统计的方法只能处理浅层的自然语言处理问题，而无法进入深层次的研究。

过去的 25 年里，随着计算能力的提高和数据量的不断增加，过去看似不可能通过统计模型完成的任务，渐渐都变得可能了，包括很复杂的句法分析。到了上个世纪 90 年代末期，大家发现通过统计得到的句法规则甚至比语言学家总结的更有说服力。2005 年后，随着 Google 基于统计方法的翻译系统全面超过基于规则方法的 SysTran 翻译系统，基于规则方法固守的最后一个堡垒被拔掉了。这才使得我们在这本书里，可以用数学的方法，而且只需要用数学的方法给出我们今天所有自然语言处理相

6 P. Brown, J Cocke, S Della Pietra, V Della Pietra, F. Jelinek, R. Mercer, P. Roossin, A statistical approach to language translation, Proceedings of the 12th conference on Computational Linguistics, P 71-76, August 22-27, 1988.

7 世界上迄今为止最成功的对冲基金，由著名微分几何数学家、陈省身-赛蒙斯定理的发明人赛蒙斯（Jim Simons）创立。彼得·布朗任 IT 部门第一主管。

关问题的全部答案。

第二点，也很有意思，基于统计的方法代替传统的方法，需要等原有的一批语言学家退休。这在科学史上也是经常发生的事。钱钟书在《围城》中讲，老科学家可以理解成“老的科学家”或者“老科学的家”两种。如果是后者，他们年纪不算老，但是已经落伍，大家必须耐心等他们退休让出位子。毕竟，不是所有人都乐意改变自己的观点，无论对错。当然，等这批人退休之后，科学就会以更快的速度发展。因此，我常想，我自己一定要在还不太糊涂和固执时就退休。



图 2.4 唉，经费又给那些“老科学家”拿走了

在科学家的新老交替上，除了贾里尼克直接领导的 IBM- 约翰 · 霍普金斯系统（包括我本人），米奇 · 马库斯（Mitch Marcus）领导的宾夕法尼亚大学也起了很大的作用。马库斯设法获得了美国自然科学基金会的支持，设立和领导了 LDC 项目，采集和整理全世界主要语言的语料，并且培养了一批世界级的科学家，让他们到世界主要的一流实验室里挑大梁。这前后两拨人形成了一个事实上的学派，占据了全世界自然语言处理学术界的主要位置。

同时，自然语言处理的应用在过去 25 年里也发生了巨大的变化。比如对自动问答的需求很大程度上被网页搜索和数据挖掘替代了。而新的应用越来越依靠数据的作用和浅层的自然语言处理的工作，这就在客观上大大加

速了自然语言处理研究从基于规则的方法到基于统计的方法的转变。

今天，几乎不再有科学家宣称自己是传统的基于规则方法的捍卫者。而自然语言处理的研究也从单纯的句法分析和语义理解，变成了非常贴近应用的机器翻译、语音识别、文本到数据库自动生成、数据挖掘和知识的获取等等。

3 小结

基于统计的自然语言处理方法，在数学模型上和通信是相通的，甚至就是相同的。因此，在数学意义上自然语言处理又和语言的初衷 —— 通信联系在一起了。但是，科学家们认识到这个联系却花了几十年的时间。

第 3 章　统计语言模型

我们在前面的章节一直强调，自然语言从它产生开始，逐渐演变成一种上下文相关的信息表达和传递的方式，因此让计算机处理自然语言，一个基本的问题就是为自然语言这种上下文相关的特性建立数学模型。这个数学模型就是在自然语言处理中常说的统计语言模型（Statistical Language Model），它是今天所有自然语言处理的基础，并且广泛应用于机器翻译、语音识别、印刷体或手写体识别、拼写纠错、汉字输入和文献查询。

1　用数学的方法描述语言规律

统计语言模型产生的初衷是为了解决语音识别问题。在语音识别中，计算机需要知道一个文字序列是否能构成一个大家理解而且有意义的句子，然后显示或者打印给使用者。

比如在上一章的例子中：

> 美联储主席本·伯南克昨天告诉媒体 7 000 亿美元的救助资金将借给上百家银行、保险公司和汽车公司。

这句话就很通顺，意思也很明白。

如果改变一些词的顺序，或者替换掉一些词，将这句话变成：

本·伯南克美联储主席昨天 7 000 亿美元的救助资金告诉媒体将借给银行、保险公司和汽车公司上百家。

意思就含混了，虽然多少还能猜到一点。

但是如果再换成：

联主美储席本·伯诉体南将借天的救克告媒昨助资金 70 元亿 00 美给上百百百家银保行、汽车险公司公司和。

基本上读者就不知所云了。

如果问一个没有学过自然语言处理的人为什么会变成这样，他可能会说，第一个句子合乎语法，词义清晰。第二个句子不合乎语法，但是词义还清晰。第三个连词义都不清晰了。20 世纪 70 年代以前，科学家们也是这样想的，试图判断这个文字序列是否合乎文法、含义是否正确等。正如我们上节所讲，这条路走不通。而贾里尼克从另外一个角度来看待这个问题，用一个简单的统计模型非常漂亮地搞定了它。

贾里尼克的出发点很简单：一个句子是否合理，就看看它的可能性大小如何。至于可能性就用概率来衡量。第一个句子出现的概率大致是 10^{-20}，第二个句子出现的概率是 10^{-25} 次方，第三个句子出现的概率是 10^{-70}。因此，第一个最有可能，它的可能是第二个句子的 10 万倍，是第三个句子的一百亿亿亿亿亿亿倍。这个方法更普通而严格的描述是：

假定 S 表示某一个有意义的句子，由一连串特定顺序排列的词 $w_1, w_2, \ldots, w_n$ 组成，这里 n 是句子的长度。现在，我们想知道 S 在文本中出现的可能性，也就是数学上所说的 S 的概率 $P(S)$。当然，可以把人类有史以来讲过的话统计一下，同时不要忘记统计进化了几百年上千年间可能讲过的话，就知道这句话可能出现的概率了。这种方法恐怕连傻子都知道行不通。因此，需要有个模型来估算它。既然 $S = w_1, w_2, \ldots, w_n$，那

么不妨把 $P(S)$ 展开表示：

$$P(S) = P(w_1, w_2, \dots, w_n) \quad (3.1)$$

利用条件概率的公式，S 这个序列出现的概率等于每一个词出现的条件概率相乘，于是 $P(w_1, w_2, \dots, w_n)$ 可展开为：

$$P(w_1, w_2, \dots, w_n) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdots P(w_n|w_1, w_2, \dots, w_{n-1}) \quad (3.2)$$

其中 $P(w_1)$ 表示第一个词 w_1 出现的概率[1]；$P(w_2|w_1)$ 是在已知第一个词的前提下，第二个词出现的概率；以此类推。不难看出，到了词 w_n，它的出现概率取决于它前面的所有词。

1 当然更准确的描述是 $P(w_1|<s>)$，即这个词在句子开头 $<s>$ 条件下的概率。

从计算上来看，第一个词的条件概率 $P(w_1)$ 很容易算，第二个词的条件概率 $P(w_2|w_1)$ 也还不太麻烦，第三个词的条件概率 $P(w_3|w_1, w_2)$ 已经非常难算了，因为它涉及到三个变量 w_1, w_2, w_3，每个变量的可能性都是一种语言字典的大小。到了最后一个词 w_n，条件概率 $P(w_n|w_1, w_2, \cdots, w_{n-1})$ 的可能性太多，无法估算。怎么办？

19 世纪到 20 世纪初，俄罗斯有个数学家叫马尔可夫（Andrey Markov），他给了个偷懒但还颇为有效的方法，也就是每当遇到这种情况时，就假设任意一个词 w_i 出现的概率只同它前面的词 w_{i-1} 有关，于是问题就变得很简单了。这种假设在数学上称为马尔可夫假设[2]。现在，S 出现的概率就变得简单了：

2 马尔可夫在 1906 年首先做出了这类过程。而将此一般化到可数无限状态空间是由柯尔莫果洛夫在 1936 年给出的。

$$P(S) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2) \cdots P(w_i|w_{i-1}) \cdots P(w_n|w_{n-1}) \quad (3.3)$$

公式（3.3）对应的统计语言模型是二元模型（Bigram Model）。顺便提一句，和语言模型相关的很多名词的中文翻译，最早是我 20 多年前提出的，依然沿用至今。Bigram Model 当初我译为二元文法模型，

现在我觉得直接叫二元模型更准确。当然，也可以假设一个词由前面 $N-1$ 个词决定，对应的模型稍微复杂些，被称为 N 元模型。我们会在下一节中介绍。

接下来的问题就是如何估计条件概率 $P(w_i|w_{i-1})$。根据它的定义：

$$P(w_i|w_{i-1})=\frac{P(w_{i-1},w_i)}{P(w_{i-1})} \tag{3.4}$$

而估计联合概率 $P(w_{i-1},w_i)$ 和边缘概率 $P(w_{i-1})$，现在变得很简单。因为有了大量机读文本，也就是专业人士讲的语料库（Corpus），只要数一数 w_{i-1},w_i 这对词在统计的文本中前后相邻出现了多少次 $\#(w_{i-1},w_i)$，以及 w_{i-1} 本身在同样的文本中出现了多少次 $\#(w_{i-1})$，然后用两个数分别除以语料库的大小 #，即可得到这些词或者二元组的相对频度：

$$f(w_{i-1},w_i)=\frac{\#(w_{i-1},w_i)}{\#} \tag{3.5}$$

$$f(w_{i-1})=\frac{\#(w_{i-1})}{\#} \tag{3.6}$$

根据大数定理，只要统计量足够，相对频度就等于概率，即

$$P(w_{i-1},w_i)\approx\frac{\#(w_{i-1},w_i)}{\#} \tag{3.7}$$

$$P(w_{i-1})\approx\frac{\#(w_{i-1})}{\#} \tag{3.8}$$

而 $P(w_i|w_{i-1})$ 就是这两个数的比值，再考虑到上面的两个概率有相同的分母，可以约掉，因此

$$P(w_i|w_{i-1})\approx\frac{\#(w_{i-1},w_i)}{\#(w_{i-1})} \tag{3.9}$$

现在，读者也许已经开始能感受到数学的美妙之处了，它把一些复杂的问题变得如此的简单。这似乎有点让人难以置信，用这么简单的数学模

型能解决复杂的语音识别、机器翻译等问题，而用很复杂的文法规则和人工智能却做不到。其实不光是普通人，就连很多语言学家都曾质疑过这种方法的有效性，但事实证明，统计语言模型比任何已知的借助某种规则的解决方法更有效。我们不妨看两个真实的例子。

在 Google 语音搜索 Google Voice 和中英文自动翻译（罗塞塔）中，发挥了最重要作用的就是这个统计语言模型。美国标准局（NIST）从 2001 年起对所有的机器翻译系统进行了评测，Google 的罗塞塔（Rosetta）系统（大家应该知道它为什么起这个名字）2007 年第一次参加 NIST 的评测，这个仅仅开发了两年的系统便一鸣惊人地夺得第一，而且评测分数高出所有基于规则的系统很多很多，要知道后者开发了十几年。这里面的秘密武器就是一个比其他竞争对手大上百倍[3]的语言模型。

3 指 N 元组的数量。

在 IBM 提出统计语言模型十几年后的 20 世纪 80 年代末，还在卡内基 - 梅隆大学做博士生的李开复用统计语言模型把 997 个词的语音识别问题简化成了一个相当于 20 个词的识别问题，实现了有史以来第一次大词汇量非特定人连续语音的识别。

当然，真正实现一个好的统计语言模型，还有许多细节问题需要解决，比如，如果上面公式中的这对词 (w_{i-1}, w_i) 在语料库中没出现，或只出现一两次，估算它的概率就有点棘手了。贾里尼克及其同事的贡献不仅在于提出了统计语言模型，而且还很漂亮地解决了所有的细节问题。这些我们将在下一节中介绍。一般的读者如果并不需要在工作中使用统计语言模型，也不想关心数学味道太浓的细节，阅读可以到此打住了，因为统计语言模型的基本原理已经介绍完了。数学的精彩之处就在于简单的模型可以干大事。

2 延伸阅读：统计语言模型的工程诀窍

读者知识背景：概率论和数理统计。

这本书的大多数章节都会有延伸阅读部分，主要写给专业读者和有兴趣学习这里面的数学原理的人，但是这些内容未必适合所有的读者。为了节省大家的时间，延伸阅读部分会标示出对读者背景知识的要求，大家可以自行决定是否跳过这部分内容。

2.1 高阶语言模型

上一节中公式（3.3）模型的假设前提是，句子中的每个词只和前面一个词有关，而和更前面的词就无关了，这似乎太简化了，或者说近似得过头了。确实是这样，读者很容易找到一些例子：某个词和前面第二个词有关，比如说“美丽的花朵”，花朵其实和美丽有关。因此，更普遍的假设是某个词和前面若干个词有关。

假定文本中的每个词 w_i 和前面 $N-1$个词有关，而与更前面的词无关，这样当前词 w_i 的概率只取决于前面 $N-1$个词 $P(w_{i-N+1},w_{i-N+2},\cdots,w_{i-1})$。因此，

$$P(w_i|w_1,w_2,\cdots,w_{i-1})=P(w_i|w_{i-N+1},w_{i-N+2},\cdots,w_{i-1}) \qquad (3.10)$$

公式（3.10）的这种假设被称为 $N-1$ 阶马尔可夫假设，对应的语言模型称为 N 元模型（N-Gram Model）。$N=2$ 的二元模型就是公式（3.3），而 $N=1$ 的一元模型实际上是一个上下文无关的模型，也就是假定当前词出现的概率与前面的词无关。而在实际应用中最多的是 $N=3$ 的三元模型，更高阶的模型就很少使用了。

为什么N 一般取值都这么小呢？这里面主要有两个原因。首先，N 元模型的大小（或者说空间复杂度）几乎是 N 的指数函数，即 $O(|V|^N)$，这里 $|V|$是一种语言词典的词汇量，一般在几万到几十万个。而使用 N 元模型

的速度（或者说时间复杂度）也几乎是一个指数函数，即 $O(|V|^{N-1})$。因此，N 不能很大。当 N 从 1 到 2，再从 2 到 3 时，模型的效果上升显著。而当模型从 3 到 4 时，效果的提升就不是很显著了，而资源的耗费增加却非常快，所以，除非是不惜资源为了做到极致，很少有人使用四元以上的模型。Google 的罗塞塔翻译系统和语言搜索系统，使用的是四元模型，该模型存储于 500 台以上的 Google 服务器中。

最后还有一个问题，是否三元或者四元甚至更高阶的模型就能覆盖所有的语言现象呢？答案显然是否定的。因为自然语言中，上下文之间的相关性可能跨度非常大，甚至可以从一个段落跨到另一个段落。因此，即使模型的阶数再提高，对这种情况也无可奈何，这就是马尔可夫假设的局限性，这时就要采用其他一些长程的依赖性（Long Distance Dependency）来解决这个问题了，在以后的章节还会有介绍。

2.2　模型的训练、零概率问题和平滑方法

使用语言模型需要知道模型中所有的条件概率，我们称之为模型的参数。通过对语料的统计，得到这些参数的过程称作模型的训练。在前一节中提到的模型训练方法，似乎非常简单。比如对于二元模型（3.3），就是拿两个数字，(w_{i-1}, w_i) 在语料中同现的次数 $\#(w_{i-1}, w_i)$ 和 (w_{i-1}) 在语料中单独出现的次数 $\#(w_{i-1})$，计算一下比值即可。但问题是，如果同现的次数 $\#(w_{i-1}, w_i) = 0$ 怎么办，是否意味着条件概率 $P(w_i|w_{i-1}) = 0$？反过来如果 $\#(w_{i-1}, w_i)$ 和 $\#(w_{i-1})$ 都只出现了一次，是否敢得出 $P(w_i|w_{i-1}) = 1$ 这样非常绝对的结论？这就涉及到统计的可靠性问题了。

在数理统计中，我们之所以敢于用对采样数据的观察结果来预测概率，是因为有大数定理（Law of Large Numbers）在背后做支持，它的要求是有足够的观测值。例如，在某镇中心的楼上，看到楼下熙熙攘攘的人群中有 550 个男性，520 个女性，我们大致可以认为这个地方男性出现的概率是 550 /（550 + 520）= 51.4%，而女性出现的概率为 520 /（550

+ 520）= 48.6%。但是，如果是一大早，我们从楼上看下去，只有 5 个人，4 个女性 1 个男性，我们是否敢说，女性出现的概率为 80%，而男性只有 20% 呢？显然不敢，因为这 5 个人出现的情况有非常大的随机性。也许第二天清晨，楼下只有 3 个人且全是男性，我们同样不敢得到女性不会出现在这里的预测。

这是生活中的常识。但是在估计语言模型的概率时，很多人恰恰忘了这个道理，因此训练出来的语言模型“不管用”，然后回过头来怀疑这个方法是否有效。其实这个方法屡试不爽，今天的数字通信很大程度就建立在这个基础上，只是如何使用的问题而已。那么如何正确地训练一个语言模型呢？

一个直接的办法就是增加数据量，但是即使如此，依然会遇到零概率或者统计量不足的问题。假定要训练一个汉语的语言模型，汉语的词汇量大致是 20 万这个量级[4]，训练一个三元模型就有 $200\,000^3 = 8 \times 10^{15}$ 个不同的参数。假如从互联网上刨去垃圾数据，有 100 亿个有意义的中文网页，这已经是相当高估的数据，每个网页平均 1 000 词。那么，即使将互联网上全部的中文内容都用作训练，依然只有 10^{13}，因此，如果用直接的比值计算概率，大部分条件概率依然是零，这种模型我们称之为“不平滑”。在实际应用中，统计语言模型的零概率问题是无法回避的，必须解决。

4 以 Google IME 为参考。

训练统计语言模型的艺术就在于解决好统计样本不足时的概率估计问题。1953 年古德（I. J. Good）在他老板图灵（Alan Turing）（就是计算机科学史上的图灵）的指导下，提出了在统计中相信可靠的统计数据，而对不可信的统计数据打折扣的一种概率估计方法，同时将折扣出来的那一小部分概率给予未看见的事件（Unseen Events）。古德和图灵还给出了一个很漂亮的重新估算概率的公式，这个公式后来被称为古德－图灵估计（Good-Turing Estimate）。

古德 - 图灵估计的原理是这样的：对于没有看见的事件，我们不能认为它发生的概率就是零，因此我们从概率的总量（Probability Mass）中，分配一个很小的比例给予这些没有看见的事件（图 3.1）。这样一来，看见的那些事件的概率总和就要小于 1 了，因此，需要将所有看见的事件概率调小一点。至于小多少，要根据"越是不可信的统计折扣越多"的方法进行。

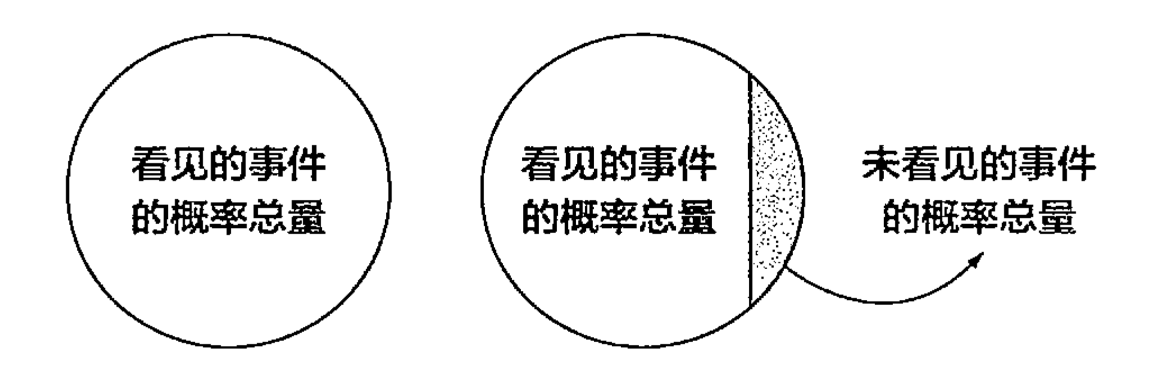


图 3.1 从左到右的变化，把一部分看得见的事件的概率分布给未看见的事件

以统计词典中的每个词的概率为例，来说明古德 - 图灵估计公式。

假定在语料库中出现 r 次的词有 N_r 个，特别地，未出现的词数量为 N_0。语料库的大小为 N。那么，很显然

$$N = \sum_{r=1}^{\infty} rN_r \quad (3.11)$$

出现 r 次的词在整个语料库中的相对频度（Relative Frequency）则是 r/N，如果不做任何优化处理，就以这个相对频度作为这些词的概率估计。

现在假定当 r 比较小时，它的统计可能不可靠，因此出现 r 次的那些词在计算它们的概率时要使用一个更小一点的次数，是 d_r（而不直接使用 r），古德 - 图灵估计按照下面的公式计算 d_r：

$$d_r = (r+1) \cdot N_{r+1}/N_r \quad (3.12)$$

显然

$$\sum_r d_r \cdot N_r = N \qquad (3.13)$$

一般来说，出现一次的词的数量比出现两次的多，出现两次的比出现三次的多。这种规律称为 Zipf 定律（Zipf's Law）。图 3.2 是一个小语料库中，出现 r 次的词数量 N_r 和 r 的关系。

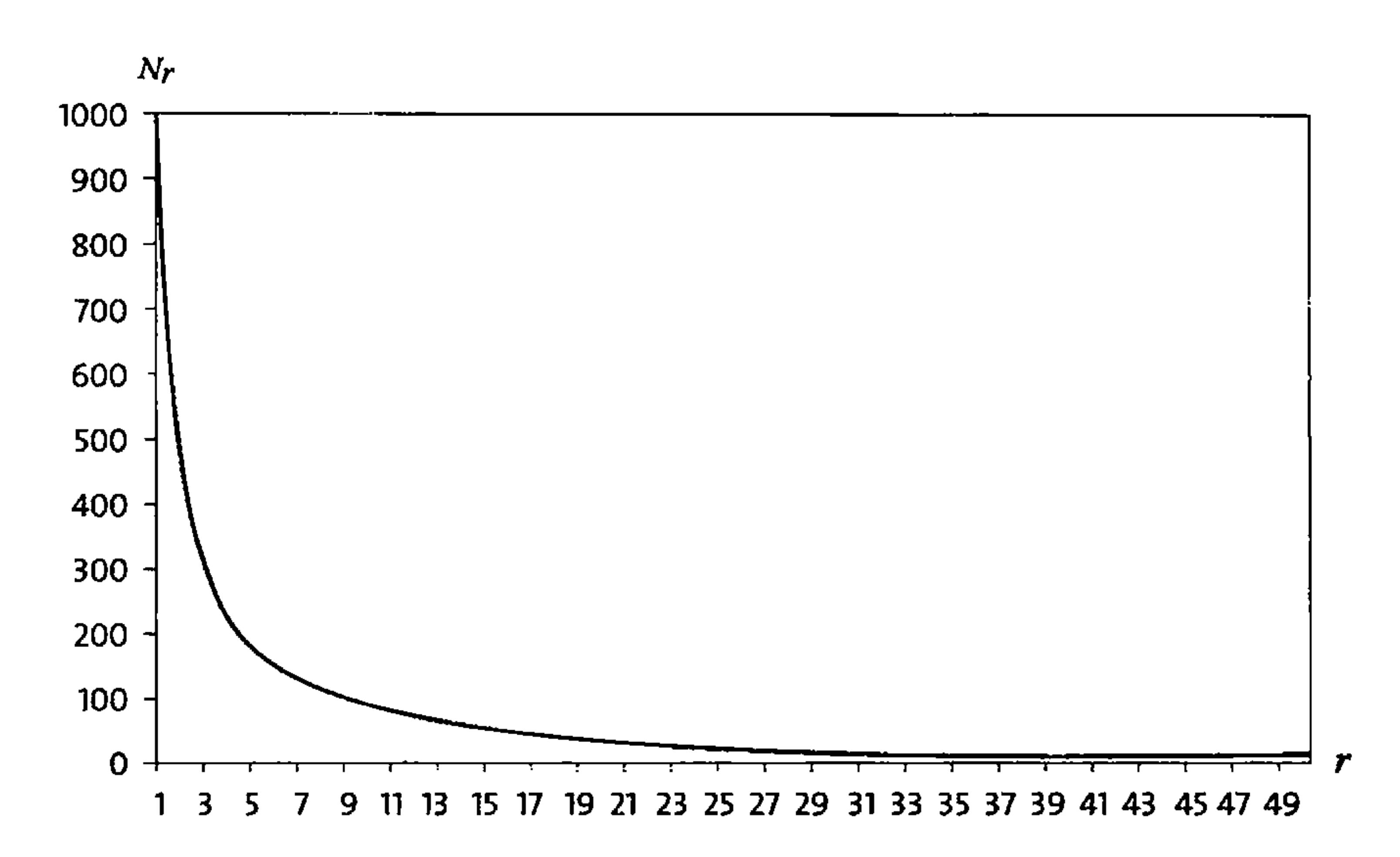


图 3.2　Zipf 定律：出现 r 次词的数量 N_r 和 r 的关系

可以看出 r 越大，词的数量 N_r 越小，即 $N_{r+1} < N_r$。因此，一般情况下 $d_r<r$，而 $d_0>0$。这样就给未出现的词赋予了一个很小的非零值，从而解决了零概率的问题。同时下调了出现频率很低的词的概率。当然，在实际的自然语言处理中，一般对出现次数超过某个阈值的词，频率不下调，只对出现次数低于这个阈值的词，频率才下调，下调得到的频率总和给未出现的词。

这样出现 r 次的词的概率估计为 d_r/N。于是，对于频率超过一定阈值的词，它们的概率估计就是它们在语料库中的相对频度，对于频率小于这个阈值的词，它们的概率估计就小于它们的相对频度，出现次数越少的，折扣越多。对于未看见的词，也给予了一个比较小的概率。这样所有词的概率估计都很平滑了。

对于二元组 (w_{i-1}, w_i) 的条件概率估计 $P(w_i|w_{i-1})$ 也可以做同样的处理。我们知道，通过前一个词 w_{i-1} 预测后一个词 w_i 时，所有的可能情况的条件概率总和应该为 1，即

$$\sum_{w_i \in V} P(w_i|w_{i-1}) = 1 \tag{3.14}$$

对于出现次数非常少的二元组 (w_{i-1}, w_i)，它们的出现次数需要按照古德－图灵的方法打折扣，这样 $\sum_{w_{i-1}, w_i \ seen} P(w_i|w_{i-1}) < 1$，同时意味着有一部分概率量没有分配出去，留给了没有看到的二元组 (w_{i-1}, w_i)。基于这种思想，估计二元模型概率的公式如下：

$$P(w_i|w_{i-1}) = \begin{cases} f(w_i|w_{i-1}) & if\ \#(w_{i-1}, w_i) \geq T \\ f_{gt}(w_i|w_{i-1}) & if\ 0 < \#(w_{i-1}, w_i) < T \\ Q(w_{i-1}) \cdot f(w_i) & otherwise \end{cases} \tag{3.15}$$

其中 T 是一个阈值，一般在 8-10 左右，函数 $f_{gt}()$ 表示经过古德－图灵估计后的相对频度，而

$$Q(w_{i-1}) = \frac{1 - \sum_{w_i \ seen} P(w_i|w_{i-1})}{\sum_{w_i \ unseen} f(w_i)} \tag{3.16}$$

这样可以保证等式（3.14）成立。

这种平滑的方法，最早由前 IBM 科学家卡茨（S. M. Katz）提出，故称为卡茨退避法（Katz backoff）。类似地，对于三元模型，概率估计的公式如下：

$$P(w_i|w_{i-2}, w_{i-1}) = \begin{cases} f(w_i|w_{i-2}, w_{i-1}) & if\ \#(w_{i-2}, w_{i-1}, w_i) \geq T \\ f_{gt}(w_i|w_{i-2}, w_{i-1}) & if\ 0 < \#(w_{i-2}, w_{i-1}, w_i) < T \\ Q(w_{i-2}, w_{i-1}) \cdot P(w_i|w_{i-1}) & otherwise \end{cases} \tag{3.17}$$

对于一般情况的 N 元模型概率估计公式，以此类推。

内伊（Herman Ney）等人在此基础上优化了 Katz backoff，原理大同小异，就不赘述了，有兴趣的读者可以读参考文献 [3.2]。

因为一元组 (w_i) 出现的次数平均比二元组 (w_{i-1}, w_i) 出现的次数要高很多，根据大数定理，它的相对频度更接近概率分布。类似地，二元组平均出现的次数比三元组要高，二元组的相对频率比三元组更接近概率分布。同时，低阶模型的零概率问题也要比高阶模型轻微点。因此，用低阶语言模型和高阶模型进行线性插值来达到平滑的目的，也是过去行业使用的一种方法，这种方法称为删除差值（Deleted Interpolation），如下面的公式。该公式中的三个λ 均为正数而且和为 1。线性插值的效果略低于卡茨退避法（Backoff），故现在已经较少使用了。

$$
\begin{aligned}
&P(w_i|w_{i-2}, w_{i-1}) \\
&\quad = \lambda(w_{i-2}, w_{i-1}) \cdot f(w_i|w_{i-2}, w_{i-1}) \\
&\quad + \lambda(w_{i-1}) \cdot f(w_i|w_{i-1}) + \lambda f(w_i) \qquad (3.18)
\end{aligned}
$$

2.3 语料的选取问题

模型训练中另一个重要的问题就是训练数据，或者说语料库的选取。如果训练语料和模型应用的领域相脱节，那么模型的效果通常要大打折扣。

比如对于建立一个语言模型，如果应用是网页搜索，它的训练数据就应该是杂乱的网页数据和用户输入的搜索串，而不是传统的、规范的新闻稿，即使前者夹杂着噪音和错误。

这里有一个很好的例子，来自于腾讯搜索部门。最早的语言模型是使用《人民日报》的语料训练的，因为开发者认为这些语料干净、无噪音。但是实际的效果就比较差，经常出现搜索串和网页不匹配的例子。后来改用网页的数据，尽管它们有很多的噪音，但是因为训练数据和应用一致，搜索质量反而好。

训练数据通常是越多越好。虽然通过上节介绍的平滑过渡的方法可以解

决零概率和很小概率的问题，但是毕竟在数据量多的时候概率模型的参数可以估计得比较准确。高阶的模型因为参数多，需要的训练数据也相应会多很多。遗憾的是，并非所有的应用都能得到足够的训练数据，比如说机器翻译的双语语料就非常少，在这种情况下片面追求高阶的大模型就变得一点意义也没有了。

在训练数据和应用数据一致并且训练量足够大的情况下，训练语料的噪音高低也会对模型的效果产生一定的影响，因此，在训练以前有时需要对训练数据进行预处理。一般情况下，少量的（没有模式的）随机噪音清除起来成本非常高，通常就不做处理了。但是对于能找到模式（Pattern）的、量比较大的噪音还是需要进行过滤的，而且它们也比较容易处理，比如网页文本中大量的制表符。因此，在成本不高的情况下，过滤训练数据还是需要做的。

3 小结

统计语言模型在形式上非常简单，任何人都很容易理解。但是里面的学问却可以很深，一个专家可以在这方面研究很多年，比如我们在延伸阅读中提到的那些问题。数学的魅力就在于将复杂的问题简单化。

参考文献：

1. Turing, Alan (October 1950), "Computing Machinery and Intelligence", Mind LIX (236): 433－460,doi:10.1093/mind/LIX.236.433

2. Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recogniser. IEEE Transactions on Acoustics, Speech, and Signal Processing, 35(3), 400－401

3. Frederick Jelinek, Statistical Methods for Speech Recognition (Language, Speech, and Communication), 1998 MIT Press.

4. Kneser, Reinhard, and Ney. 1995. Improved backing-off for m-gram language modeling. In Proceedings of ICASSP-95, vol. 1,18－184.

第 4 章　谈谈中文分词

1　中文分词方法的演变

在第 3 章中我们谈到可以利用统计语言模型进行自然语言处理，而这些语言模型是建立在词的基础上的，因为词是表达语义的最小单位。对于西方拼音语言来讲，词之间有明确的分界符（Delimit），统计和使用语言模型非常直接。而对于中、日、韩、泰等语言，词之间没有明确的分界符（韩语名词短语和动词之间有分界符，但是短语内没有）。因此，首先需要对句子进行分词，才能做进一步的自然语言处理。

若分词的输入是一串胡子连着眉毛的汉字，例如一个句子：中国航天官员应邀到美国与太空总署官员开会。而分词的输出是用分界符，比如用斜线或者竖线分割的一串词：中国 / 航天 / 官员 / 应邀 / 到 / 美国 / 与 / 太空 / 总署 / 官员 / 开会。

最容易想到的分词方法，也是最简单的办法，就是查字典。这种方法最早是由北京航天航空大学的梁南元教授提出的。“查字典”的办法，其实就是把一个句子从左向右扫描一遍，遇到字典里有的词就标识出来，遇到复合词（比如“上海大学”）就找最长的词匹配，遇到不认识的字串就分割成单字词，于是简单的分词就完成了。这种简单的分词方法完全能处理上面例子中的句子。当我们从左到右扫描时，先遇到“中”这个字，

它本身是一个单字词，我们可以在这里做一个切割，但是，当我们再遇到“国”字时，发现它可以和前面的“中”字组成一个更长的词，因此，我们就将分割点放在“中国”的后面。接下来，我们发现“中国”不会和后面的字组成更长的词，这个分割点就最终确定了。

这个最简单的方法可以解决七八成以上的分词问题，应该讲它在复杂性（成本）不高的前提下，取得了还算满意的效果。但是，它毕竟太简单，遇到稍微复杂一点的问题就无能为力了。20 世纪 80 年代，哈尔滨工业大学的王晓龙博士把查字典的方法理论化，发展成最少词数的分词理论，即一句话应该分成数量最少的词串。这种方法一个明显的不足是当遇到有二义性（有双重理解意思）的分割时就无能为力了。比如，对短语“发展中国家”，正确的分割是“发展 - 中 - 国家”，而从左向右查字典的办法会将它分割成“发展 - 中国 - 家”，显然是错了。另外，并非所有的最长匹配都一定是正确的。比如“上海大学城书店”的正确分词应该是“上海 - 大学城 - 书店”，而不是“上海大学 - 城 - 书店”，“北京大学生”的正确分词是“北京 - 大学生”，而不是“北京大学 - 生”。

我们在第一章介绍过，语言中的歧义性伴随着语言的发展，困扰了学者们上千年。在中国古代，断句和说文解字从根本上讲，就是消除歧义性，而不同学者之间的看法也显然不相同。各种春秋的正义或者对论语的注释，就是各家按照自己的理解消除歧义性。分词的二义性是语言歧义性的一部分。20 世纪 90 年代以前，海内外不少学者试图用一些文法规则来解决分词的二义性问题，都不是很成功。当然也有一些学者开始注意到统计信息的作用，但是并没有找到有完善理论基础的正确方法。1990 年前后，当时在清华大学电子工程系工作的郭进博士用统计语言模型成功解决了分词二义性问题，将汉语分词的错误率降低了一个数量级。上面举的二义性的例子用统计语言模型都可以解决。

郭进是中国大陆自觉地用统计语言模型方法进行自然语言处理的第一人，并且获得了成功。这里面除了他比较努力以外，他特殊的经历也起了很

大作用。他和我一样，虽然是计算机专业的博士，但是却在以通信为主的科系工作，他周围的同事都是长期从事通信研究的，可以说是贾里尼克的同行，因此，他在方法论上接受通信模型非常直接。

利用统计语言模型分词的方法，可以用几个数学公式简单概括如下：假定一个句子 S 可以有几种分词方法，为了简单起见，假定有以下三种：

$$A_1, A_2, A_3, \cdots, A_k$$

$$B_1, B_2, B_3, \cdots, B_m$$

$$C_1, C_2, C_3, \cdots, C_n$$

其中，$A_1, A_2 \ldots B_1, B_2, \ldots, C_1, C_2 \ldots$ 等等都是汉语的词，上述各种分词结果可能产生不同数量的词串，因为我用了 k, m, n 三个不同的下标表示句子在不同的分词时词的数目。那么最好的一种分词方法应该保证分完词后这个句子出现的概率最大。也就是说，如果 $A_1, A_2, A_3, \cdots, A_k$ 是最好的分法，那么其概率满足

$$P(A_1, A_2, A_3, \cdots, A_k) > P(B_1, B_2, \cdots, B_m)$$

并且

$$P(A_1, A_2, A_3, \cdots, A_k) > P(C_1, C_2, \cdots, C_n)$$

因此，只要利用上一章提到的统计语言模型计算出每种分词后句子出现的概率，并找出其中概率最大的，就能够找到最好的分词方法。

当然，这里面有一个实现的技巧。如果穷举所有可能的分词方法并计算出每种可能性下句子的概率，那么计算量是相当大的。因此，可以把它看成是一个动态规划（Dynamic Programming）的问题，并利用维特比（Viterbi）算法快速地找到最佳分词。（我们在后面的章节会介绍该算法。）上述过程可以用下图来描述：

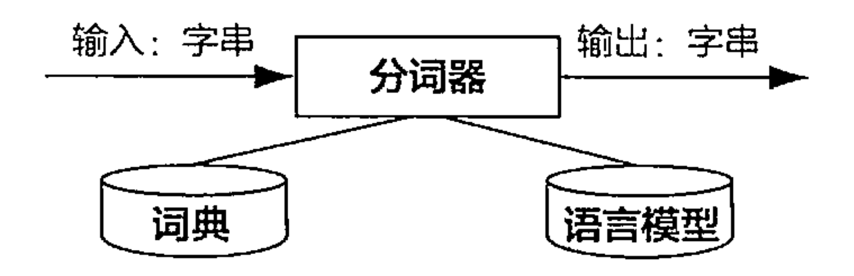


图 4.1 分词器示意图

在郭进博士之后，海内外不少学者利用统计的方法，进一步完善了中文分词。其中值得一提的是清华大学孙茂松教授和香港科技大学吴德凯教授的工作（见参考文献）。孙茂松教授的贡献主要在于解决没有词典的情况下的分词问题，而吴德凯教授是较早将中文分词方法用于英文词组的分割，并且将英文词组和中文词在机器翻译时对应起来。

需要指出的是，语言学家对词语的定义不完全相同。比如说“北京大学”，有人认为是一个词，而有人认为该分成两个词。一个折中的解决办法是在分词的同时，找到复合词的嵌套结构。在上面的例子中，如果一句话包含“北京大学”四个字，那么先把它当成一个四字词，然后再进一步找出细分词“北京”和“大学”。这种方法最早是郭进在 *Computational Linguistics*（《计算机语言学》）杂志上发表的，此后不少系统都采用这种方法。

一般来讲，根据不同应用，汉语分词的颗粒度大小应该不同。比如，在机器翻译中，颗粒度应该大一些，“北京大学”就不能被分成两个词。而在语音识别中，“北京大学”一般是被分成两个词。因此，不同的应用应该有不同的分词系统。Google 早期由于工程师少，没有精力开发，只能直接使用 Basis Technology 公司的通用分词器，分词结果没有针对搜索进行优化。因此，后来 Google 有两位工程师葛显平博士和朱安博士，专门为搜索设计和实现了自己的分词系统，以适应搜索特殊的需求。

在不少人看来，分词技术只是针对亚洲语言的，而罗马体系的拼音语言没有这个问题，其实不然。也许大家想不到，中文分词的方法也被应用到英语处理，主要是手写体识别中。因为在识别手写体时，单词之间的

空格就不很清楚了。中文分词方法可以帮助判别英语单词的边界。

其实，语言处理的许多数学方法是通用的，和具体的语言无关。在 Google 内部，我们在设计语言处理的算法时，都会考虑它是否能很容易地适用于各种自然语言。这样才能有效地支持上百种语言的搜索。

最后，需要指出的是任何方法都有它的局限性，虽然利用统计语言模型进行分词，可以取得比人工更好的结果，但是也不可能做到百分之百准确。因为统计语言模型很大程度上是依照“大众的想法”，或者“多数句子的用法”，而在特定情况下可能是错的。另外，有些人为创造出的“两难”的句子，比如对联“此地安能居住，其人好不悲伤”[1]，使用什么方法也无法消除二义性。好在真实文本中，这些情况几乎不会发生。

1 它的两种分词方法“此地 - 安能 - 居住，其人 - 好不 - 悲伤”和“此地安 - 能居住，其人好 - 不悲伤”意思完全相反。

2　延伸阅读：工程上的细节问题

2.1　分词的一致性

如何衡量分词结果的对与错，好与坏看似容易，其实不是那么简单。说它看似容易，是因为只要对计算机分词的结果和人工分词的结果进行比较就可以了。说它不是那么简单，是因为不同的人对同一个句子可能有不同的分词方法。比如有的人认为“清华大学”应该是一个词，有的人却认为“清华大学”是一个复合词，应该分成“清华 - 大学”两个单词。应该讲，这两种分法都有道理，虽然语言学家可能比较坚持某一种是正确的。在不同的应用中，经常是一种词的切分比另一种更有效。

不同的人对词的切分看法上的差异性远比我们想象的要大得多。1994 年，我和 IBM 的研究人员合作，对此进行了研究。IBM 提供了 100 个有代表性的中文整句，我组织 30 名清华大学二年级的本科生独立地对它们进行分词。实验前，为了保证大家对词的看法基本一致，我们对 30 名学生进行了半个小时的培训。实验结果表明，这 30 名教育水平相当的大学生分词的一致性只有 85%-90%。

在将统计语言模型用于分词以前，分词的准确率通常较低，可以提升的空间非常大。不同的人切分的区别虽然会影响分词评测的准确性，但是好的方法和坏的方法还是可以根据分词结果与人工切分的比较来衡量的。

当统计语言模型被广泛应用后，不同的分词器产生的结果的差异要远远小于不同人之间看法的差异，这时简单依靠与人工分词的结果比较来衡量分词器的准确性就很难，甚至是毫无意义的了。很难讲一个准确率在 97%的分词器就一定比另一个准确率为 95% 的要好，因为这要看它们选用的所谓正确的人工分词的数据是如何得来的。我们甚至只能讲某个分词器和另一个分词器相比，与人工分词结果的吻合度稍微高一点而已。所幸的是，中文分词现在是一个已经解决了的问题，提高的空间微乎其微了。只要采用统计语言模型，效果都差不到哪里去。

2.2 词的颗粒度和层次

人工分词产生不一致性的原因主要在于人们对词的颗粒度的认识问题。在汉语里，词是表达意思最基本的单位，再小意思就变了。这就如同在化学里分子是保持化学性质的最小单位一样。再往下分到原子，化学的特性就变了。在这一点上所有的语言学家都没有异议。因此，对于“贾里尼克”这个词，所有的语言学家都会认为它不可拆分，如果拆成四个字，和原来的人名就没有联系了。但是对于“清华大学”，大家的看法就不同了，有些人认为它是一个整体，表示北京西郊一所特定的大学，也有人认为它是一个词组，或者说是一个名词短语，“清华”是修饰“大学”的定语，因此需要拆开，不拆开就无法体现它里面的修饰关系。这就涉及到对词的颗粒度的理解了。

在这里不去强调谁的观点对，而是要指出在不同的应用中，会有一种颗粒度比另一种更好的情况。比如在机器翻译中，一般来讲，颗粒度大翻译效果好。比如“联想公司”作为一个整体，很容易找到它对应的英语翻译 Lenovo，如果分词时将它们分开，很有可能翻译失败，因为联想在

汉语中，首先是“根据相关联的场景想象”的意思。但是在另外一些应用，比如网页搜索中，小的颗粒度比大的颗粒度要好。比如清华大学这四个字如果作为一个词，在对网页分词后，它是一个整体了，当用户查询“清华”时，是找不到清华大学的，这绝对是有问题的。

虽然可以对不同的应用构造不同的分词器，但是这样做不仅非常浪费，而且也不必要。更好的方法是让一个分词器同时支持不同层次的词的切分。也就是说，上面的“清华大学”既可以被看成一个整体，也可以被切分开，然后由不同的应用自行决定采用哪个颗粒度的切分。这在原理和实现上并不是很麻烦，简单介绍如下。

首先需要一个基本的词表和一个复合词的词表。基本词表包括像“清华”、“大学”、“贾里尼克”这样无法再分的词。复合词表包含复合词以及它们由哪些基本词构成，包括像“清华大学：清华 - 大学”，“搜索引擎：搜索 - 引擎”。

接下来需要根据基本词表和复合词表各建立一个语言模型，比如 L1 和 L2。

然后根据基本词表和语言模型 L1 对句子进行分词，就得到了小颗粒度的分词结果。如果对应到图 4.1 的分词器中，这里面的字串是输入，词串是输出。顺便讲一句，基本词比较稳定，分词方法又是解决了的，因此小颗粒度的分词除了偶尔增加点新词外，没有什么需要研究和做的工作。

最后，在此基础上，再用复合词表和语言模型 L2 进行第二次分词，对于图 4.1 的分词器，这时输入是基本词串，输出是复合词串，词表和语言模型两个数据库改变了，但是分词器（程序）本身和前面的完全相同。

介绍了分词的层次概念后，我们可以更进一步讨论分词器准确性的问题了。分词的不一致性可以分为错误和颗粒度不一致两种，错误又分成两类，一类是越界型错误，比如把“北京大学生”分成“北京大学 - 生”。另一类是覆盖型错误，比如把“贾里尼克”拆成了四个字。这些是明显的

错误，是改进分词器时要尽可能消除的。接下来是颗粒度的不一致性，人工分词的不一致性大多属于此类。这一类不一致性在度量分词器的好坏时，可以不作为错误，以免不同人看法的不同左右了对分词器的度量。对于某些应用，需要尽可能地找到各种复合词，而不是将其切分。因此，需要花一些功夫做数据挖掘的工作，不断完善复合词的词典（它的增长速度较快），这是近年来中文分词主要花精力的地方。

3 小结

中文分词以统计语言模型为基础，经过几十年的发展和完善，今天基本上可以看做是一个已经解决的问题。

当然不同的人做的分词器有好有坏，这里面的差别主要在于数据的使用和工程实现的精度。

参考文献：

1. 梁南元 书面汉语自动分词系统
http://www.touchwrite.com/demo/LiangNanyuan-JCIP-1987.pdf
2. 郭进 统计语言模型和汉语音字转换的一些新结果
http://www.touchwrite.com/demo/GuoJin-JCIP-1993.pdf
3. 郭进
Critical Tokenization and its Properties http://acl.ldc.upenn.edu/J/J97/J97-4004.pdf
4. 孙茂松
Chinese word segmentation without using lexicon and hand-crafted training data
http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=980775
5. Dekai WU. "Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora". IJCAI-95: 14th Intl. Joint Conf. on Artificial Intelligence, 1328-1335. Montreal: Aug 1995

第 5 章　隐含马尔可夫模型

隐含马尔可夫模型是一个并不复杂的数学模型，到目前为止，它一直被认为是解决大多数自然语言处理问题最为快速、有效的方法。它成功地解决了复杂的语音识别、机器翻译等问题。当我们看完这些复杂的问题是如何通过简单的模型描述和解决时，会不得不由衷地感叹数学模型之妙。

1　通信模型

我们在第一、二章中介绍了，人类信息交流的发展贯穿了人类的进化和文明的全过程。而自然语言是人类交流信息的工具，语言和通信的联系是天然的。通信的本质就是一个编解码和传输的过程。但是自然语言处理早期的努力都集中在语法、语义和知识表述上，离通信的原理越走越远，而这样离答案也就越来越远。当自然语言处理的问题回归到通信系统中的解码问题时，很多难题都迎刃而解了。

让我们先来看一个典型的通信系统：当一个人（或者机器）发送信息时，他需要采用一种能在媒体中（比如空气、电线）传播的信号，比如语音或者电话线的调制信号，这个过程是广义上的编码。然后通过媒体传播到接收方，这个过程是信道传输。在接收方，收听的人（或者机器）根据事先约定好的方法，将这些信号还原成发送者的信息，这个过程是广

义上的解码。下图表示了一个典型的通信系统，它包含雅格布森（Roman Jakobson）提出的通信的六个要素[1]。

1 雅格布森通信六个要素是：发送者（信息源），信道，接收者，信息，上下文和编码。

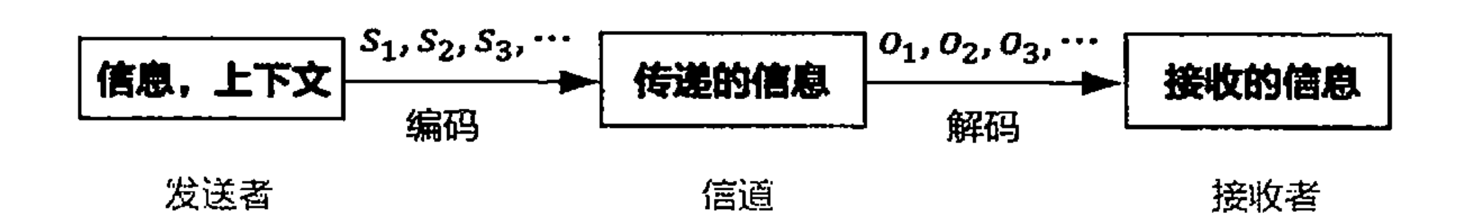


图 5.1 通信模型

其中$s_1, s_2, s_3, \cdots$表示信息源发出的信号，比如手机发送的信号。$o_1, o_2, o_3, \cdots$是接收器（比如另一部手机）接收到的信号。通信中的解码就是根据接收到的信号$o_1, o_2, o_3, \cdots$还原出发送的信号$s_1, s_2, s_3, \cdots$。

那么这与自然语言处理的工作，比如语音识别，又有什么直接的关系呢？不妨换一个角度来考虑这个问题。所谓语音识别，就是听话的人去猜测说话者要表达的意思。这其实就像通信中，根据接收端收到的信号去分析、理解、还原发送端传送过来的信息。我们平时在说话时，脑子就是一个信息源。我们的喉咙（声带）、空气，就是如电线和光缆般的信道。听众的耳朵就是接收器，而听到的声音就是传送过来的信号。根据声学信号来推测说话者的意思，就是语音识别。如果接收端是一台计算机而不是人，那么计算机要做的就是语音的自动识别。

同样，很多自然语言处理的应用也可以这样理解。在从汉语到英语的翻译中，说话者讲的是汉语，但是信道传播编码的方式是英语，如果利用计算机，根据接收到的英语信息，推测说话者的汉语意思，就是机器翻译。同样，如果要根据带有拼写错误的语句推测说话者想表达的正确意思，那就是自动纠错。这样，几乎所有的自然语言处理问题都可以等价成通信的解码问题。

在通信中，如何根据接收端的观测信号$o_1, o_2, o_3, \cdots$来推测信号源发送的信息$s_1, s_2, s_3, \cdots$呢？只需要从所有的源信息中找到最可能产生出观测信

号的那一个信息。用概率论的语言来描述，就是在已知 $o_1,o_2,o_3,\cdots$ 的情况下，求得令条件概率

$P(s_1,s_2,s_3,\cdots|o_1,o_2,o_3,\cdots)$达到最大值的那个信息串$s_1,s_2,s_3,\cdots$，即

$$s_1,s_2,s_3,\cdots=\underset{all\ s_1,s_2,s_3,\cdots}{\mathrm{Arg\,Max}}\,P(s_1,s_2,s_3,\cdots|o_1,o_2,o_3,\cdots) \quad (5.1)$$

其中 Arg 是参数 Argument 的缩写，表示能获得最大值的那个信息串。当然，上面的概率不容易直接求出，不过可以间接地计算它。利用贝叶斯公式可以把上述公式等价变换成

$$\frac{P(o_1,o_2,o_3,\cdots|s_1,s_2,s_3,\cdots)\cdot P(s_1,s_2,s_3,\cdots)}{P(o_1,o_2,o_3,\cdots)} \quad (5.2)$$

其中 $P(o_1,o_2,o_3,\cdots|s_1,s_2,s_3,\cdots)$ 表示信息 $s_1,s_2,s_3,\cdots$ 在传输后变成接收的信号 $o_1,o_2,o_3,\cdots$ 的可能性；而 $P(s_1,s_2,s_3,\cdots)$ 表示 $s_1,s_2,s_3,\cdots$ 本身是一个在接收端合乎情理的信号（比如一个合乎情理的句子）的可能性；最后 $P(o_1,o_2,o_3,\cdots)$ 表示在发送端（比如说话的人）产生信息 $o_1,o_2,o_3,\cdots$ 的可能性。

大家读到这里也许会问，你现在是不是把问题变得更复杂了，因为公式越写越长了。别着急，我们现在就来简化这个问题。首先，一旦信息 $o_1,o_2,o_3,\cdots$ 产生了，它就不会改变了，这时 $P(o_1,o_2,o_3,\cdots)$ 就是一个可以忽略的常数。因此，上面的公式可以等价成

$$P(o_1,o_2,o_3,\cdots|s_1,s_2,s_3,\cdots)\cdot P(s_1,s_2,s_3,\cdots) \quad (5.3)$$

当然，这里面还有两项，虽然多过（5.1）的一项，但是这个公式完全可以用隐含马尔可夫模型（Hidden Markov Model）来估计。

2 隐含马尔可夫模型



图 5.2 俄罗斯著名科学家安德烈・马尔可夫

隐含马尔可夫模型（Hidden Markov Model）其实并不是 19 世纪俄罗斯数学家马尔可夫(Andrey Markov)发明的，而是美国数学家鲍姆(Leonard E. Baum）等人在 20 世纪六七十年代发表的一系列论文中提出的，隐含马尔可夫模型的训练方法（鲍姆 - 韦尔奇算法）也是以他的名字命名的。

要介绍隐含马尔可夫模型，还是要从马尔可夫链说起。到了 19 世纪，概率论的发展从对（相对静态的）随机变量的研究发展到对随机变量的时间序列 $s_1, s_2, s_3, \cdots, s_t, \cdots$，即随机过程（动态的）的研究。这在哲学的意义上，是人类认识的一个飞跃。但是，随机过程要比随机变量复杂得多。首先，在任何一个时刻 t，对应的状态 s_t 都是随机的。举一个大家熟悉的例子，我们可以把 $s_1, s_2, s_3, \cdots, s_t, \cdots$ 看成是北京每天的最高气温，这里面每个状态 s_t 都是随机的。第二，任何一个状态 s_t 的取值都可能和周围其他的状态相关。回到上面的例子，任何一天的最高气温，与这段时间以前的最高气温是相关的。这样随机过程就有两个维度的不确定性。马尔可夫为了简化问题，提出了一种简化的假设，即随机过程中各个状态 s_t 的概率分布，只与它的前一个状态 s_{t-1} 有关，即 $P(s_t|s_1, s_2, s_3, \cdots, s_{t-1}) = P(s_t|s_{t-1})$。比如，对于天气预报，硬性假定今天的气温只与昨天有关而和前天无关。当然这种假设未必适合所有的应用，但是至少对以前很多不好解决的问题给出了近似解。这个假设后来被命名为马尔可夫假设，而符合这个假设的随机过程则称为马尔可夫过程，也称为马尔可夫链。下图表示一个离散的马尔可夫过程。

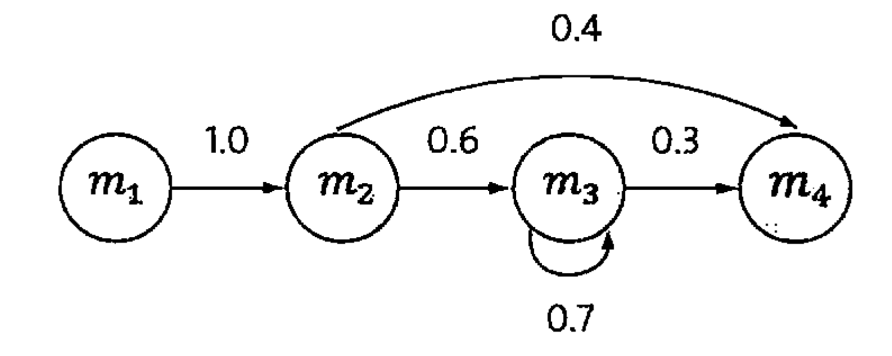


图 5.3 马尔可夫链

在这个马尔可夫链中，四个圈表示四个状态，每条边表示一个可能的状态转换，边上的权值是转移概率。例如，状态 m_1 到 m_2 之间只有一条边，且边上权值为 1.0。这表示从状态 m_1 只可能转换到状态 m_2，转移概率为 1.0。从 m_2 出发的有两条边：到 m_3 和到 m_4。其中权值 0.6 表示：如果某个时刻 t 的状态 s_t 是 m_2，则下一个时刻的状态 $s_{t+1} = m_3$ 的概率（可能性）是 60%。如果用数学符号表示是 $P(s_{t+1} = m_3|s_t = m_2) = 0.6$。类似的，有 $P(s_{t+1} = m_4|s_t = m_2) = 0.4$。

把这个马尔可夫链想象成一台机器，它随机地选择一个状态作为初始状态，随后按照上述规则随机选择后续状态。这样运行一段时间 T 之后，就会产生一个状态序列：$s_1, s_2, s_3, \cdots s_T$。看到这个序列的人，不难数出某个状态 m_i 的出现次数 $\#(m_i)$，以及从 m_i 转换到 m_j 的次数 $\#(m_i, m_j)$，从而估计出从 m_i 到 m_j 的转移概率 $\#(m_i, m_j)/\#(m_i)$。每一个状态只和前面一个有关，比如从状态 3 到状态 4，不论在此之前是如何进入到状态 3 的（是从状态 2 进入，还是在状态 3 本身转了几个圈子），这个概率都是 0.3。

隐含马尔可夫模型是上述马尔可夫链的一个扩展：任一时刻 t 的状态 s_t 是不可见的。所以观察者没法通过观察到一个状态序列 $s_1, s_2, s_3, \cdots s_T$ 来推测转移概率等参数。但是，隐含马尔可夫模型在每个时刻 t 会输出一个符号 o_t，而且 o_t 和 s_t 相关且仅和 s_t 相关。这个被称为独立输出假设。隐含马尔可夫模型的结构如下：其中隐含的状态 $s_1, s_2, s_3, \cdots$ 是一个典型的马尔可夫链。鲍姆把这种模型称为“隐含”马尔可夫模型。

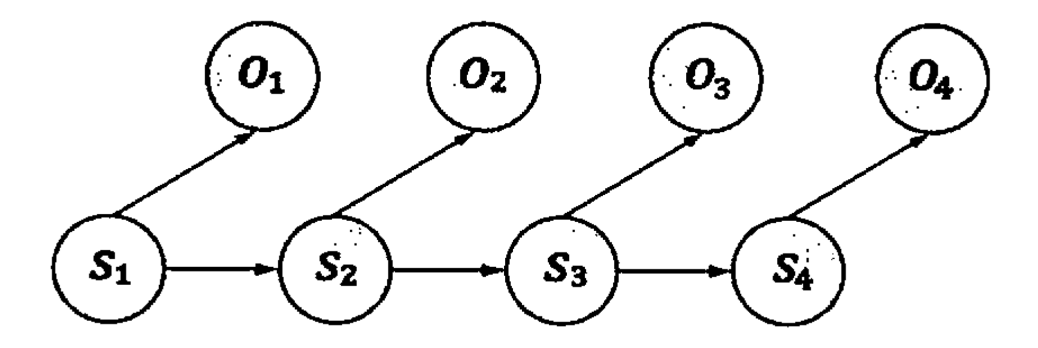


图 5.4 隐含马尔可夫模型

基于马尔可夫假设和独立输出假设，我们可以计算出某个特定的状态序列 $s_1,s_2,s_3,\cdots$ 产生出输出符号 $o_1,o_2,o_3,\cdots$ 的概率。

$$P(s_1,s_2,s_3,\cdots,o_1,o_2,o_3,\cdots)=\prod_t P(s_t|s_{t-1})\cdot P(o_t|s_t) \quad (5.4)$$

读者可能已经看出，公式（5.4）在形态上和公式（5.3）非常相似。现在我们把马尔可夫假设和独立输出假设用于通信的解码问题（5.3），即把

$$\begin{aligned}P(o_1,o_2,o_3,\cdots\mid s_1,s_2,s_3,\cdots)&=\prod_t P(o_t|s_t)\\P(s_1,s_2,s_3,\cdots)&=\prod_t P(s_t|s_{t-1})\end{aligned} \quad (5.5)$$

代人（5.3），这时正好得到（5.4）。这样通信的解码问题就可以用隐含马尔可夫模型来解决了。而很多自然语言处理问题是和通信的解码问题等价的，因此它们完全可以由隐含马尔可夫模型来解决。至于如何找出上面式子的最大值，进而找出要识别的句子 $s_1,s_2,s_3,\cdots$，可以利用维特比算法（Viterbi Algorithm），这里面的细节我们会在后面的章节中介绍。

在公式（5.3）中，$P(s_1,s_2,s_3,\cdots)$是语言模型，我们在前面的一章已经介绍过了。

$P(s_1,s_2,s_3,\cdots|o_1,o_2,o_3,\cdots)$ 根据应用的不同而有不同的名称，在语音识别中它被称为“声学模型”（Acoustic Model），在机器翻译中是“翻译模型”(Translation Model),而在拼写校正中是“纠错模型”(Correction Model）。

隐含马尔可夫模型成功的应用最早是语音识别。20 世纪 70 年代，当时

IBM 华生实验室的贾里尼克领导的科学家们，主要是刚刚从卡内基 - 梅隆大学毕业的贝克夫妇（James and Janet Baker）[2]，提出用隐含马尔可夫模型来识别语音，语音识别的错误率相比人工智能和模式匹配等方法降低了三分之二（从 30% 到 10%）。20 世纪 80 年代末李开复博士坚持采用隐含马尔可夫模型的框架，成功地开发了世界上第一个大词汇量连续语音识别系统 Sphinx。接下来，隐含马尔可夫模型陆续成功地应用于机器翻译、拼写纠错、手写体识别、图像处理、基因序列分析等很多 IT 领域，近 20 年来，它还广泛应用于股票预测和投资。

2
李开复的师兄和师姐，后来共同创立了 Dragon 语言公司，现已离异。

我最早接触到隐含马尔可夫模型是20多年前的事。那时在“随机过程”（清华过去“臭名昭著”的一门课）里学到这个模型，但当时实在想不出它有什么实际用途。几年后，我在清华跟随王作英教授学习、研究语音识别时，他给了我几十篇文献。我印象最深的就是贾里尼克和李开复的文章，它们的核心思想就是隐含马尔可夫模型。复杂的语音识别问题居然能如此简单地表述、解决，我由衷地感叹数学模型之妙。

3 延伸阅读：隐含马尔可夫模型的训练

读者知识背景：概率论。

围绕着隐含马尔可夫模型有三个基本问题：

1. 给定一个模型，如何计算某个特定的输出序列的概率；
2. 给定一个模型和某个特定的输出序列，如何找到最可能产生这个输出的状态序列；
3. 给定足够量的观测数据，如何估计隐含马尔可夫模型的参数。

第一个问题相对简单，对应的算法是 Forward-Backward 算法，在此略过，有兴趣的读者可以参看弗里德里克·贾里尼克（Frederick Jelinek）的 *Statistical Methods for Speech Recognition*（*Language, Speech, and Communication*）一书[3]。第二个问题可以用著名的维特比算法解决，我们

3
The MIT Press (January 16, 1998)

在以后的章节中会介绍。第三个问题就是我们这一节要讨论的模型训练问题。

在利用隐含马尔可夫模型解决实际问题中，需要事先知道从前一个状态 s_{t-1} 进入当前状态 s_t 的概率 $P(s_t|s_{t-1})$，也称为转移概率（Transition Probability），和每个状态 s_t 产生相应输出符号 o_t 的概率 $P(o_t|s_t)$，也称为生成概率（Generation Probability）。这些概率被称为隐含马尔可夫模型的参数，而计算或者估计这些参数的过程称为模型的训练。

我们从条件概率的定义出发，知道：

$$P(o_t|s_t) = \frac{P(o_t, s_t)}{P(s_t)} \tag{5.6}$$

$$P(s_t|s_{t-1}) = \frac{P(s_{t-1}, s_t)}{P(s_{t-1})} \tag{5.7}$$

对于公式（5.6）的状态输出概率，如果有足够多人工标记（Human Annotated）的数据，知道经过状态 s_t 有多少次 $\#(s_t)$，每次经过这个状态时，分别产生的输出 o_t 是什么，而且分别有多少次 $\#(o_t, s_t)$ 就可以用两者的比值

$$P(o_t|s_t) \approx \frac{\#(o_t, s_t)}{\#(s_t)} \tag{5.8}$$

直接算出（估计出）模型的参数。因为数据是人工标注的，因此这种方法称为有监督的训练方法（Supervised Training）。对于公式（5.7）的转移概率，其实和前面提到的训练统计语言模型的条件概率是完全相同的，因此可以依照统计语言模型的训练方法

$$P(w_i|w_{i-1}) \approx \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})} \tag{5.9}$$

直接得到。有监督的训练的前提是需要大量人工标注的数据。很遗憾的是，很多应用都不可能做到这件事，比如在语言识别中的声学模型训练，人是无法确定产生某个语音的状态序列的，因此也就无法标注训练模型的数据。而在另外一些应用中，虽然标注数据是可行的，但是成本非常高。比如训练中英机器翻译的模型，需要大量中英对照的语料，还要把中英

文的词组一一对应起来，这个成本非常高。因此，训练隐含马尔可夫模型更实用的方式是仅仅通过大量观测到的信号 $o_1, o_2, o_3, \cdots$ 就能推算模型参数的 $P(s_t|s_{t-1})$ 和 $P(o_t|s_t)$ 的方法，这类方法称为无监督的训练算法，其中主要使用的是鲍姆 - 韦尔奇算法（Baum-Welch Algorithm）。

两个不同的隐含马尔可夫模型可以产生同样的输出信号，因此，仅仅通过观察到的输出信号来倒推产生它的隐含马尔可夫模型可能会得到很多个合适的。但是总会是一个模型 $M_{\theta 2}$ 比另一个 $M_{\theta 1}$ 更有可能产生观测到的输出，其中 θ_2 和 θ_1 是隐含马尔可夫模型的参数。鲍姆 - 韦尔奇算法就是寻找这个最可能的模型 $M_{\hat{\theta}}$。

在鲍姆 - 韦尔奇算法的思想是这样的：

首先找到一组能够产生输出序列 O 的模型参数。（显然它们是一定存在的，因为转移概率 P 和输出概率 Q 为均匀分布时，模型可以产生任何输出，当然包括我们观察到的输出 O。）现在，有了这样一个初始的模型，我们称为 $M_{\theta 0}$，需要在此基础上找到一个更好的模型。假定解决了第一个问题和第二个问题，不但可以算出这个模型产生 O 的概率 $P(O|M_{\theta 0})$，而且能够找到这个模型产生 O 的所有可能的路径以及这些路径的概率。这些可能的路径，实际上记录了每个状态经历的多少次，到达了哪些状态，输出了哪些符号，因此可以将它们看做是“标注的训练数据”，并且根据公式（5.6）和（5.7）计算出一组新的模型参数 θ_1，从 $M_{\theta 0}$ 到 $M_{\theta 1}$ 的过程称为一次迭代。可以证明

$$P(O|M_{\theta 1}) > P(O|M_{\theta 0}) \qquad (5.10)$$

接下来，我们从 $M_{\theta 1}$ 出发，可以找到一个更好的模型 $M_{\theta 2}$，并且不断地找下去，直到模型的质量没有明显提高为止。这就是鲍姆 - 韦尔奇算法的原理，对于具体算法的公式，有兴趣的读者可以阅读参考文献 [5.2]，我就不再赘述了。

鲍姆 - 韦尔奇算法的每一次迭代都是不断地估计（Expectation）新的模

型参数，使得输出的概率（我们的目标函数）达到最大化（Maximization），因此这个过程被称为期望值最大化（Expectation-Maximization），简称 EM 过程。EM 过程保证算法一定能收敛到一个局部最优点，很遗憾它一般不能保证找到全局最优点。因此，在一些自然语言处理的应用中，比如词性标注，这种无监督的鲍姆-韦尔奇算法训练出的模型比有监督的训练得到的模型效果略差，因为前者未必能收敛到全局最优点。但是如果目标函数是凸函数（比如信息熵），则只有一个最优点，在这种情况下 EM 过程可以找到最佳值。

4 小结

隐含马尔可夫模型最初应用于通信领域，继而推广到语音和语言处理中，成为连接自然语言处理和通信的桥梁。同时，隐含马尔可夫模型也是机器学习主要工具之一。和几乎所有的机器学习的模型工具一样，它需要一个训练算法（鲍姆-韦尔奇算法）和使用时的解码算法（维特比算法），掌握了这两类算法，就基本上可以使用隐含马尔可夫模型这个工具了。

参考文献：

1. Baum, L. E.; Petrie, T. (1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". The Annals of Mathematical Statistics 37 (6): 1554-1563.

2. Baum, L. E.; Eagon, J. A. (1967). "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology". Bulletin of the American Mathematical Society 73 (3):

3. Baum, L. E.; Sell, G. R. (1968). "Growth transformations for functions on manifolds". Pacific Journal of Mathematics 27 (2): 211-227.

4. Baum, L. E.; Petrie, T.; Soules, G.; Weiss, N. (1970). "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". The Annals of Mathematical Statistics 41:

5. Jelinek, F.; Bahl, L.; Mercer, R. (1975). "Design of a linguistic statistical decoder for the recognition of continuous speech". IEEE Transactions on Information Theory 21 (3): 250.

第 6 章　信息的度量和作用

到目前为止，虽然我们一直在谈论信息，但是信息这个概念依然有些抽象。我们常常说信息很多，或者信息较少，但却很难说清楚信息到底有多少。比如一本 50 多万字的中文书《史记》到底有多少信息量，或者一套莎士比亚全集有多少信息量。我们也常说信息有用，那么它的作用是如何客观、定量地体现出来的呢？信息用途的背后是否有理论基础呢？这两个问题几千年来都没有人给出很好的解答。直到 1948 年，香农（Claude Shannon）在他著名的论文“通信的数学原理”（*A Mathematic Theory of Communication*）中提出了“信息熵”（读 shāng)）的概念，才解决了信息的度量问题，并且量化出信息的作用。

1　信息熵

一条信息的信息量和它的不确定性有着直接的关系。比如说，我们要搞清楚一件非常非常不确定的事，或是我们一无所知的事情，就需要了解大量的信息。相反，如果我们对某件事已经有了较多的了解，那么不需要太多的信息就能把它搞清楚。所以，从这个角度来看，可以认为，信息量就等于不确定性的多少。

那么如何量化信息的度量呢？来看一个例子。2010 年举行了世界杯足球赛，大家都很关心谁会是冠军。假如我错过了看世界杯，赛后我问一个知道比赛结果的观众“哪支球队是冠军”？他不愿意直接告诉我，而让我猜，并且我每猜一次，他要收一元钱才肯告诉我是否猜对了，那么我需要付给他多少钱才能知道谁是冠军呢？我可以把球队编上号，从 1 到 32，然后提问：“冠军的球队在 1-16 号中吗？”假如他告诉我猜对了，我会接着问：“冠军在 1-8 号中吗？”假如他告诉我猜错了，我自然知道冠军队在 9-16 号中。这样只需要五次，我就能知道哪支球队是冠军。所以，谁是世界杯冠军这条消息的信息量只值 5 块钱。

当然，香农不是用钱，而是用“比特”（Bit）这个概念来度量信息量。一个比特是一位二进制数，计算机中的一个字节是 8 比特。在上面的例子中，这条消息的信息量是 5 比特。（如果有朝一日有 64 支球队进入决赛阶段的比赛，那么“谁是世界杯冠军”的信息量就是 6 比特，因为要多猜一次。）读者可能已经发现，信息量的比特数和所有可能情况的对数函数 log 有关[1]。（log32 = 5，log64 = 6）

1 如无特别说明，本书中的对数一律以 2 为底。

有些读者此时可能会发现实际上可能不需要猜五次就能猜出谁是冠军，因为像西班牙、巴西、德国、意大利这样的球队得冠军的可能性比日本、南非、韩国等球队大得多。因此，第一次猜测时不需要把 32 支球队等分成两个组，而可以把少数几支最可能的球队分成一组，把其他队分成另一组。然后猜冠军球队是否在那几支热门队中。重复这样的过程，根据夺冠概率对剩下的候选球队分组，直至找到冠军队。这样，也许三次或四次就猜出结果。因此，当每支球队夺冠的可能性（概率）不等时，“谁是世界杯冠军”的信息量比 5 比特少。香农指出，它的准确信息量应该是

$$H=-(p_1\cdot\log p_1+p_2\cdot\log p_2+\cdots+p_{32}\cdot\log p_{32}) \qquad (6.1)$$

其中，$p_1,p_2,\ldots,p_{32}$ 分别是这 32 支球队夺冠的概率。香农把它称为“信息熵”（Entropy），一般用符号 H 表示，单位是比特。有兴趣的读者可以推算一下当 32 支球队夺冠概率相同时，对应的信息熵等于 5 比特。有数

学基础的读者还可以证明上面公式的值不可能大于 5。对于任意一个随机变量 X（比如得冠军的球队），它的熵定义如下：

$$H(X) = -\sum_{x \in X} P(x) \log P(x) \tag{6.2}$$

变量的不确定性越大，熵也就越大，把它搞清楚所需要的信息量也就越大。信息量的量化度量为什么叫做“熵”这么一个奇怪的名字呢？因为它的定义形式和热力学的熵有很大的相似性。这一点我们在扩展阅读中再介绍。

有了“熵”这个概念，就可以回答本文开始提出的问题，即一本 50 万字的中文书平均有多少信息量。我们知道，常用的汉字（一级二级国标）大约有 7 000 字。假如每个字等概率，那么大约需要 13 比特（即 13 位二进制数）表示一个汉字。但汉字的使用是不平衡的。实际上，前 10% 的汉字占常用文本的 95% 以上。因此，即使不考虑上下文的相关性，而只考虑每个汉字的独立概率，那么，每个汉字的信息熵大约也只有 8-9 比特。如果再考虑上下文相关性，每个汉字的信息熵就只有 5 比特左右。所以，一本 50 万字的中文书，信息量大约是 250 万比特。如果用一个好的算法压缩一下，整本书可以存成一个 320KB 的文件。如果直接用两字节的国标编码存储这本书，大约需要 1MB 大小，是压缩文件的三倍。这两个数量的差距，在信息论中称作“冗余度”（Redundancy）。需要指出的是这里讲的 250 万比特是个平均数，同样长度的书，所含的信息量可以相差很多。如果一本书重复的内容很多，它的信息量就小，冗余度就大。

不同语言的冗余度差别很大，而汉语在所有语言中冗余度是相对小的。大家可能都有这个经验，一本英文书，翻译成汉语，如果字体大小相同，那么中译本一般都会薄很多。这和人们普遍的认识 —— 汉语是最简洁的语言 —— 是一致的。对中文信息熵有兴趣的读者可以读我和王作英教授在《电子学报》上合写的一篇文章“汉语信息熵和语言模型的复杂度”[2]。

2 http://engine.cqvip.com/content/citation.dll?id=2155540

2 信息的作用

自古以来，信息和消除不确定性是相联系的。在英语里，信息和情报是同一个词（Information），而我们知道情报的作用就是排除不确定性。有些时候，在战争中一比特的信息能抵过千军万马。在第二次世界大战中，当纳粹德国兵临前苏联莫斯科城下时，斯大林在欧洲已经无兵可派，而他们在西伯利亚的中苏边界却有60万大军不敢使用，因为苏联人不知道德国的轴心国盟友日本当时的军事策略是北上进攻前苏联，还是南下和美国开战。如果是南下，那么苏联人就可以放心大胆地从亚洲撤回60万大军增援莫斯科会战。事实上日本人选择了南下，其直接行动是后来的偷袭珍珠港，但是苏联人并不知晓。斯大林不能猜，因为猜错了后果是很严重的。这个“猜”既是指扔钢镚儿似的卜卦，也包括主观的臆断。最后，传奇间谍佐尔格向莫斯科发去了信息量仅1比特、却价值无限的情报（信息）：“日本将南下”，于是前苏联就把西伯利亚所有的军队调往了欧洲战场。后面的故事大家都知道了。

如果把这个故事背后的信息论原理抽象化、普遍化，可以总结如下：

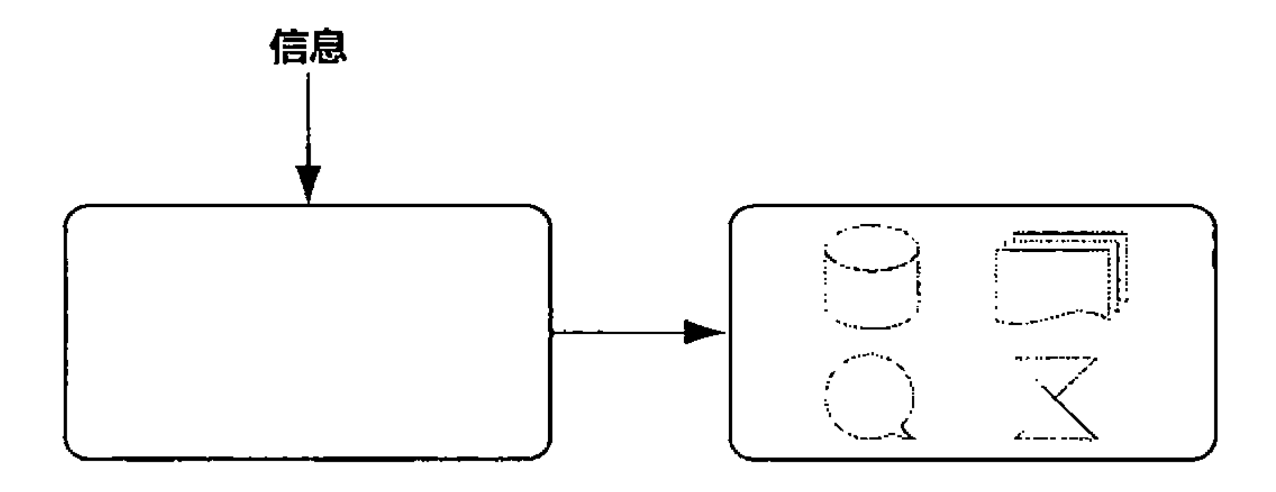


图6.1 信息是消除系统不确定性的唯一办法（在没有获得任何信息前，一个系统就像是一个黑盒子，引入信息，就可以了解黑盒子系统的内部结构）

一个事物（比如上面讲到的日本内阁的战略决定）内部会存在着随机性，也就是不确定性，假定为U，而从外部消除这个不确定性唯一的办法是引入信息I，而需要引入的信息量取决于这个不确定性的大小，即$I > U$才行。当$I < U$时，这些信息可以消除一部分不确定性，也就是说新的不确定性。

$$U' = U - I \qquad (6.3)$$

反之，如果没有信息，任何公式或者数字的游戏都无法排除不确定性。这个朴素的结论非常重要，但是在研究工作中经常被一些半瓶子醋的专家忽视，希望做这方面工作的读者谨记。几乎所有的自然语言处理、信息与信号处理的应用都是一个消除不确定性的过程。读了这本书早期博客的读者很多都反映，希望我多讲点搜索方面的例子，因此这里就以搜索为例说明信息的作用。

网页搜索本质上就是要从大量（几十亿个）网页中，找到和用户输入的搜索词最相关的几个网页。几十亿个可能性，当然是很大的不确定性 U。如果只剩下几个网页，就几乎没有了不确定性了（此时 $U' << U$），甚至是完全确定了（对于导航类搜索就是如此，第一条结果通常就是要找的网页）。因此，网页搜索本质上也是利用信息消除不确定性的过程。如果提供的信息不够多，比如搜索词是常用的关键词，诸如“中国”、“经济”之类的，那么会有好多相关的结果，用户可能还是无从选择。这时正确的做法是挖掘新的隐含的信息，比如网页本身的质量信息。如果这些信息还是不够消除不确定性，不妨再问问用户。这就是相关搜索的理论基础。不正确的做法是在这个关键词上玩数字和公式的游戏，由于没有额外的信息引入，这种做法没有效果，这就是很多做搜索质量的人非常辛苦却很少有收获的原因。最糟糕的做法是引入人为的假设，这和“蒙”没什么差别。其结果是似乎满足了个别用户的口味，但是对大部分用户来讲，搜索结果反而变得更糟（这就如同斯大林胡乱猜测日本的战略意图一样）。合理利用信息，而不是玩弄什么公式和机器学习算法，是做好搜索的关键。

知道的信息越多，随机事件的不确定性就越小。这些信息，可以是直接针对我们要了解的随机事件，比如上面提到的日本内阁的战略决定；也可以是和我们关心的随机事件相关的其他（事件）的信息 —— 通过获取这些相关信息也能帮助我们了解所关注的对象。比如在前面几章提到的自然语言的统计模型，其中的一元模型就是通过某个词本身的概率分布，来消除不确定因素；而二元及更高阶的语言模型则还使用了上下文的信息，那就能准确预测一个句子中当前的词汇了。在数学上可以严格地证

明为什么这些“相关的”信息也能够消除不确定性。为此，需要引入一个条件熵（Conditional Entropy）的概念。

假定 X 和 Y 是两个随机变量，X 是我们需要了解的。假定我们现在知道了 X 的随机分布 $P(X)$，那么也就知道了 X 的熵：

$$H(X)=-\sum_{x\in X}P(x)\cdot\log P(x) \qquad (6.4)$$

那么它的不确定性就是这么大。现在假定我们还知道 Y 的一些情况，包括它和 X 一起出现的概率，在数学上称为联合概率分布（Joint Probability），以及在 Y 取不同值的前提下 X 的概率分布，在数学上称为条件概率分布（Conditional Probability）。定义在 Y 的条件下 X 的条件熵为：

$$H(X|Y)=-\sum_{x\in X,y\in Y}P(x,y)\log P(x|y) \qquad (6.5)$$

在本章的延伸阅读中，我们会证明 $H(X)\geq H(X|Y)$，也就是说多了 Y 的信息，关于 X 的不确定性下降了！在统计语言模型中，如果把 Y 看成是前一个字，那么在数学上就证明了二元模型的不确定性小于一元模型。同样的道理，可以定义有两个条件的条件熵

$$H(X|Y,Z)=-\sum_{x\in X,y\in Y,z\in Z}P(x,y,z)\log P(x|y,z) \qquad (6.6)$$

还可以证明 $H(X|Y)\geq H(X|Y,Z)$。也就是说，三元模型应该比二元的好。

最后还有一个有意思的问题：在上面的公式中的等号什么时候成立？等号成立说明增加了信息，不确定性却没有降低。这可能么？答案是肯定的，即当我们获取的信息和我们要研究的事物没有关系的时候。再回到本节上面的例子，如果佐尔格送去的情报是关于德国人和英国人在北非的军事行动，则不论这样的情报有多少，都解决不了斯大林的困惑。

用一句话概括这一节：信息的作用在于消除不确定性，自然语言处理的大量问题就是找相关的信息。

3 延伸阅读：信息论在信息处理中的应用

读者知识背景：概率论。

前面已经介绍了信息熵，它是信息论的基础，信息论在自然语言处理中扮演着指导性的角色，在这一节里我们看看信息论的另外两个重要的概念——“互信息”（Mutual Information）和“相对熵”（Relative Entropy，或 Kullback-Leibler Divergence），以及它们在自然语言处理中的作用。

3.1 互信息

前面 6.2 节中提到，当获取的信息和要研究的事物“有关系”时，这些信息才能帮助我们消除不确定性。当然“有关系”这种说法太模糊，太不科学，最好能够量化地度量“相关性”。比如常识告诉我们，一个随机事件“今天北京下雨”和另一个随机变量“过去 24 小时北京空气的湿度”的相关性就很大，但是它们的相关性到底有多大？再比如“过去 24 小时北京空气的湿度”似乎就和“旧金山的天气”相关性不大，但我们是否能说它们毫无相关性[3]？为此，香农在信息论中提出了一个“互信息”的概念作为对两个随机事件“相关性”的量化度量。

3 按照蝴蝶效应的理论，它们的相关性并不如想象得那么小！

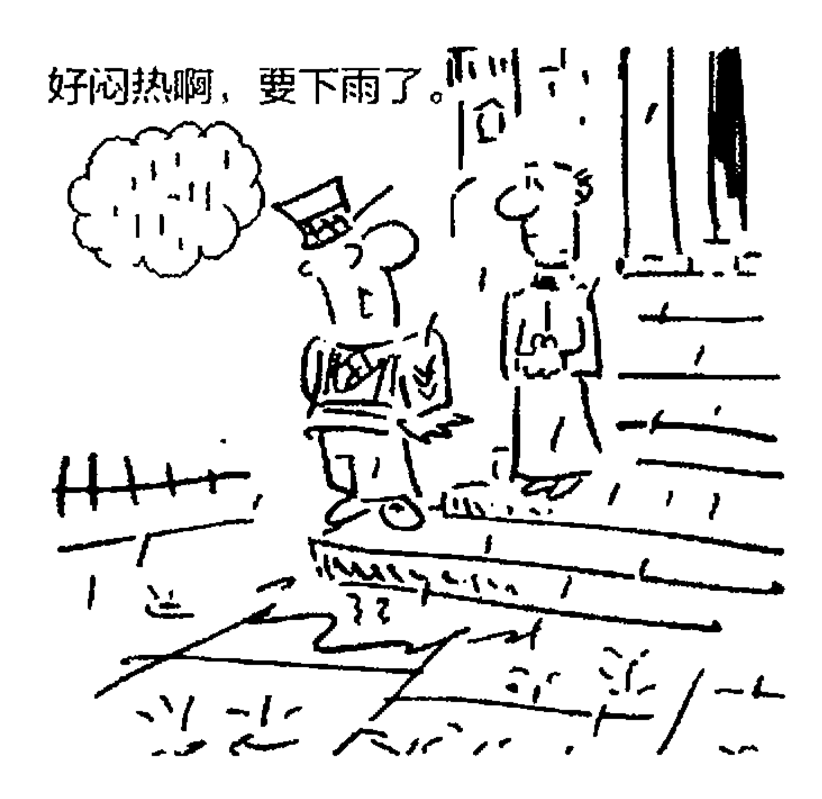


图 6.2 好闷热啊，要下雨了。闷热和下雨直接的互信息很高

假定有两个随机事件 X 和 Y，它们的互信息定义如下：

$$I(X;Y)=\sum_{x\in X,y\in Y}P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \quad (6.7)$$

很多见了公式就头大的读者看了这个定义可能会感到有点烦。不过没关系，大家只要记住这个符号 $I(X;Y)$ 就好。我们接着会证明，其实这个互信息就是上节介绍的随机事件 X 的不确定性或者说熵 $H(X)$，以及在知道随机事件 Y 条件下的不确定性，或者说条件熵 $H(X|Y)$ 之间的差异，即

$$I(X;Y)=H(X)-H(X|Y) \quad (6.8)$$

现在清楚了，所谓两个事件相关性的量化度量，就是在了解了其中一个 Y 的前提下，对消除另一个 X 不确定性所提供的信息量。需要讲一下，互信息是一个取值在 0 到 $\min(H(X), H(Y))$ 之间的函数，当 X 和 Y 完全相关时，它的取值是 1；当二者完全无关时，它的取值是 0。

在自然语言处理中，两个随机事件，或者语言特征的互信息是很容易计算的。只要有足够的语料，就不难估计出互信息公式中的 $P(X,Y)$，$P(X)$ 和 $P(Y)$ 三个概率，进而算出互信息。因此，互信息被广泛用于度量一些语言现象的相关性。

机器翻译中，最难的两个问题之一是词义的二义性（又称歧义性，Ambiguation）问题。比如 Bush 一词可以是美国总统布什的名字，也可以是灌木丛。（有一个笑话，2004 年和布什争夺总统的民主党候选人克里的名字 Kerry 被一些机器翻译系统翻译成了“爱尔兰的小母牛”，这是 Kerry 在英语中的另外一个意思。）




图 6.3 布什和克里电视辩论（“灌木丛”总统，“小母牛”参议员）

那么如何正确地翻译这些词呢？人们很容易想到要用语法、要分析语句等等。其实，迄今为止，没有一种语法能很好地解决这个问题，因为 Bush 不论翻译成人名还是灌木丛，都是名词，在语法上没有太大问题。当然爱较真的读者可能会提出“必须加上总统做宾语时，主语得是一个人这条规则”，要是这样，语法规则就多得数不清了，而且还有很多例外，比如一个国家在国际组织中也可以做主席（总统）的轮值国。其实，真正简单却非常实用的方法是使用互信息。具体的解决办法大致如下：首先从大量文本中找出和总统布什一起出现的互信息最大的一些词，比如总统、美国、国会、华盛顿等等，当然，再用同样的方法找出和灌木丛一起出现的互信息最大的词，比如土壤、植物、野生等等。有了这两组词，在翻译 Bush 时，看看上下文中哪类相关的词多就可以了。这种方法最初是由吉尔（William Gale）、丘奇（Kenneth Church）和雅让斯基（David Yarowsky）提出的。

20 世纪 90 年代初，雅让斯基在宾夕法尼亚大学是自然语言处理大师马库斯（Mitch Marcus）教授的博士生，他很多时候都泡在贝尔实验室丘奇等人的研究室里。也许是急于毕业，他在吉尔等人的帮助下想出了一个最快也是最好地解决翻译中的二义性的方法，就是上面的方法，这个看上去简单的方法效果好得让同行们大吃一惊。雅让斯基因而只花了三年就从马库斯那里拿到了博士，而他的师兄弟们平均要花六年时间。

3.2 相对熵

信息论中另外一个重要的概念是"相对熵"，在有些文献中它被称为"交叉熵"，在英语中是 Kullback-Leibler Divergence，是以它的两个提出者库尔贝克和莱伯勒的名字命名的。相对熵也用来衡量相关性，但和变量的互信息不同，它用来衡量两个取值为正数的函数的相似性，它的定义如下：

$$KL(f(x)\|g(x)) = \sum_{x\in X} f(x)\cdot\log\frac{f(x)}{g(x)} \qquad (6.9)$$

同样，大家不必关心公式本身，只要记住下面三条结论就好：

1. 对于两个完全相同的函数，它们的相对熵等于零。
2. 相对熵越大，两个函数差异越大；反之，相对熵越小，两个函数差异越小。
3. 对于概率分布或者概率密度函数，如果取值均大于零，相对熵可以度量两个随机分布的差异性。

在自然语言处理中相对熵的应用很多，比如用来衡量两个常用词（在语法上和语义上）在不同文本中的概率分布，看它们是否同义；或者根据两篇文章中不同词的分布，看看它们的内容是否相近等等。利用相对熵，可以得到信息检索中最重要的一个概念：词频率－逆向文档频率（TF-IDF），后面会在网页搜索相关性和新闻分类中进一步介绍TF-IDF的概念。

熵、条件熵和相对熵这三个概念与语言模型的关系非常密切。我们在第二章中谈到语言模型时，没有讲如何定量地衡量一个语言模型的好坏，因为当时还没有介绍这三个概念。当然，读者会很自然地想到，既然语言模型能减少语音识别和机器翻译的错误，那么就拿一个语音识别系统或者机器翻译软件来试试，好的语言模型必然导致错误率较低。这种想法是对的，而且今天的语音识别和机器翻译也是这么做的。但这种测试方法对于研发语言模型的人来讲，既不直接，又不方便，而且很难从错

误率反过来定量度量语言模型。事实上，在贾里尼克等人研究语言模型时，世界上既没有像样的语音识别系统，更没有机器翻译。我们知道，语言模型是为了用上下文预测当前的文字，模型越好，预测得越准，那么当前文字的不确定性就越小。

信息熵正是对不确定性的衡量，因此可以想象信息熵能直接用于衡量统计语言模型的好坏。当然，因为有了上下文的条件，所以对高阶的语言模型，应该用条件熵。如果再考虑到从训练语料和真实应用的文本中得到的概率函数有偏差，就需要再引入相对熵的概念。贾里尼克从条件熵和相对熵出发，定义了一个称为语言模型复杂度（Perplexity）的概念，直接衡量语言模型的好坏。复杂度有很清晰的物理含义，它是在给定上下文的条件下，句子中每个位置平均可以选择的单词数量。一个模型的复杂度越小，每个位置的词就越确定，模型越好。

李开复博士在介绍他发明的 Sphinx 语音识别系统的论文里谈到，如果不用任何语言模型（即零元语言模型）时，复杂度为 997，也就是说句子中每个位置有 997 个可能的单词可以填入。如果（二元）语言模型只考虑前后词的搭配不考虑搭配的概率时，复杂度为 60。虽然它比不用语言模型好很多，但是和考虑了搭配概率的二元语言模型相比要差很多，因为后者的复杂度只有 20。

对信息论有兴趣又有一定数学基础的读者，可以阅读斯坦福大学托马斯·科弗（Thomas Cover）教授的专著《信息论基础》（*Elements of Information Theory*）。科弗教授是当今最权威的信息论专家。

4 小结

信息熵不仅是对信息的量化度量，而且是整个信息论的基础。它对于通信、数据压缩、自然语言处理都有很强的指导意义。信息熵的物理含义是对一个信息系统不确定性的度量，在这一点上，它和热力学中熵的概念相同，

因为后者是对于一个系统无序的度量。这说明科学上很多看似不同的学科之间也会有很强的相似性。

参考文献：

1. Thomas M. Cover, Joy A. Thomas. *Elements of Information Theory* New York: Wiley, 1991. ISBN 0-471-06259-6

中译本：

Thomas M. Cover, Joy A. Thomas，信息论基础，清华大学出版社，2003

2. Kai-Fu Lee, Automatic Speech Recognition:The Development of the SPHINX System, Springer, 1989.

3. Gale, W., K. Church, and D. Yarowsky. ``A Method for Disambiguating Word Senses in a Large Corpus." Computers and the Humanities. 26, pp. 415-439, 1992

第 7 章 贾里尼克和现代语言处理

谨以本章纪念弗里德里克·贾里尼克博士
1932 年 11 月 18 日 – 2010 年 9 月 14 日

当我最初在"谷歌黑板报"上发表"数学之美"系列文章时，为了引起读者的兴趣，介绍了一些成功地将数学原理应用于自然语言处理领域的大师和学者。但我的根本目的不是为了单纯讲故事，更不是为了聊八卦，而是为了给有志于信息领域研究的年轻人介绍一批大师和成功者，让大家学习到他们的思维方法，从而能获得他们那样的成功。在当今物欲横流的中国社会，学术界浮躁，年轻人浮躁，少数有着远大志向的年轻人实际上是非常孤独的。这很像罗曼·罗兰描写一战后的法国。罗曼·罗兰为那些追求灵魂高尚而非物质富裕的年轻人写下了《巨人三传》[1]，让大家呼吸到巨人的气息。今天，我希望把一批大师介绍给有志学子。我们从弗里德里克·贾里尼克开始。

1 即《贝多芬传》、《米开朗基罗传》和《托尔斯泰传》。

按顺序读到这一章的读者也许注意到了，我们在前面的章节中多次提到了贾里尼克这个名字。事实上，现代语音识别和自然语言处理确实是跟他的名字紧密联系在一起的。在这里我不想列举他的贡献，而想讲一讲他作为一个普通人的故事。这些事要么是我亲身经历的，要么是他亲口对我讲的。



图 7.1 贾里尼克

1 早年生活

弗里德里克·贾里尼克（Frederek Jelinek，我们称他弗莱德）出生于捷克克拉德诺（Kladno）[2]一个富有的犹太家庭，他的父亲是一位牙科医生。承袭了犹太民族的传统，弗莱德的父母从小就很注意他的教育，并且打算送他去英国的公学（私立学校）读书。为了教他学好德语，还专门请了一位德国的家庭女教师。但是第二次世界大战完全打碎了他们的梦想。他们先是被从家中赶了出去，流浪到布拉格。他的父亲死在了集中营，弗莱德自己成天在街上玩耍，完全荒废了学业。二战后，当弗莱德再度回到学校时，他不仅要从小学补起，而且成绩一塌糊涂，全部是 D，但是很快他就赶上了班上的同学。不过，他在小学时从来没有得过 A。

2 捷克中部距首都布拉格 25 公里的小城。

1946 年，捷克开始了前苏联似的集权统治。弗莱德的母亲吸取了他父亲当年的教训，果断决定带着并不富有的全家移民美国。弗莱德后来讲，"我母亲做了一个非常正确的决定（指离开捷克到美国一事），她没有犯我和父亲同样的错误。当年我父亲已经把所有的（牙医）设备运到了英国，可是他对德国人还是抱有幻想，在最后时刻留了下来。"在美国，贾里尼克一家生活非常贫困，全家基本是靠母亲做点心赚钱为生，弗莱德当时只有十几岁，就进工厂打工赚钱补贴家用。显然，他没有（可能）

天天呆在教室和家里，把时间都花在课本上，他在上大学前花在读书上的时间恐怕连现在一般好学生的一半都不到。当然，我自己在小学（文革阶段）和中学（上个世纪 80 年代）花在课本上的时间也不到现在学生的一半。所以我们都不赞同中小学生只会上学考试的教育方式。

每当弗莱德和我谈起我们各自少年时的教育，我们都同意这样几个观点。首先，小学生和中学生其实没有必要花那么多时间读书，而他们的社会经验、生活能力以及在那时树立起的志向将帮助他们的一生。第二，中学阶段花很多时间比同伴多读的课程，在大学以后用非常短的时间就可以读完，因为在大学阶段，人的理解力要强得多。举个例子，在中学需要花 500 小时才能学会的内容，在大学可能花 100 小时就够了。因此，一个学生在中小学阶段建立的那一点点优势在大学很快就会丧失殆尽。第三，学习（和教育）是一个人一辈子的过程，很多中学成绩好的亚裔学生进入名校后表现明显不如那些因为兴趣而读书的美国同伴，因为前者不断读书的动力不足。第四，书本的内容可以早学，也可以晚学，但是错过了成长阶段却是无法补回来的。（因此，少年班的做法不足取。）现在中国的好学校里，恐怕百分之九十九的孩子在读书上花的时间比我当时要多，更比贾里尼克要多得多，但是这些孩子今天可能有百分之九十九在学术上的建树不如我，更不如贾里尼克。这实在是教育的误区。

贾里尼克最初的理想在他十来岁时就建立起来了，他原本想成为一个律师，为他父亲那样的冤屈者辩护，但是到美国后，他很快意识到他那浓厚的外国口音将使他在法庭上的辩护很吃力。贾里尼克的第二个理想是成为医生，也算是子承父业。他想进哈佛大学医学院，但他无力承担医学院 8 年高昂的学费（4 年的本科教育加上 4 年的医学院教育）。而恰恰此时麻省理工学院给了他一份（为东欧移民设的）全额奖学金。贾里尼克决定到麻省理工学电机工程。贾里尼克的理想在不断改变，但是他通过努力走向成功的志向一直没有改变。

在那里，他遇到了许多世界级的大师，包括信息论的鼻祖香农博士，和

语言学大师雅格布森（Roman Jakobson，他提出了著名的通信六要素[3]）。后来贾里尼克的太太米兰娜从捷克来到美国，在哈佛大学求学，弗莱德经常去邻校哈佛陪着太太听课。在那里，他经常去听伟大的语言学家乔姆斯基（Noam Chomsky）的课。这三位大师对贾里尼克后来的研究方向——利用信息论解决语言问题产生了重要影响。我一直认为，一个人想要在自己的领域做到世界一流，他的周围必须有非常多的一流人物。贾里尼克的幸运之处在于他在年轻的时候得到了这些大师的指点，以后在研究境界上比同龄人高出了一筹。

3 雅格布森的通信模型见第三章。

弗莱德从麻省理工获得博士学位后，在哈佛大学教了一年书，然后到康奈尔大学任教，成了贾里尼克教授。他之所以选择康奈尔大学，是因为找工作时和那里的一位语言学家哈克特（Charles Hackott）谈得颇为投机。当时那位教授表示愿意和贾里尼克在利用信息论解决语言问题上进行合作。但是，等贾里尼克到康奈尔以后，那位教授表示对语言学不再有兴趣而转向写歌剧了。贾里尼克对语言学家的坏印象从此开始。加上后来他在 IBM 时发现语言学家们嘴上头头是道，干起活来高不成低不就，对语言学家从此深恶痛绝。他甚至说："我每开除一名语言学家，我的语音识别系统识别率就会提高一点。"[4] 这句话后来在业界广为流传，为每一个搞语音识别和语言处理的人所熟知。

4 "Every time I fire a linguist, the performance of the speech recognizer goes up".

2 从水门事件到莫妮卡·莱温斯基

这个标题不是我为了哗众取宠而起的，而是贾里尼克在 1999 年 ICASSP[5] 做的大会报告的题目[6]，因为水门事件发生的时间（1972 年）恰恰是统计语音识别和自然语言处理开始的时间，而因莱温斯基事件弹劾克林顿总统也正好发生于当时会议前一年。

5 国际声学、语音和信号处理大会，International Conference on Acoustic, Speech and Signal Processing

6 http://icassp99.asu.edu/technical/plenary/jelinek.html

贾里尼克在康奈尔十年磨一剑，潜心研究信息论，终于悟出了自然语言处理的真谛。1972 年，贾里尼克到 IBM 华生实验室做学术休假（Sabbatical），无意中领导了语音识别实验室，两年后他在康奈尔和 IBM 之间选择了留

在 IBM。在那里，贾里尼克组建的研究队伍阵容之强大可谓空前绝后，其中包括他的著名搭档波尔（L. Bahl），著名的语音识别 Dragon 公司的创始人贝克夫妇（Jim Baker & Janet Baker），解决最大熵迭代算法的达拉皮垂（S. Della Pietra 和 V. Della Pietra）孪生兄弟，BCJR 算法的另外两个共同提出者库克（J. Cocke）和拉维夫（J. Raviv），以及第一个提出机器翻译统计模型的布朗（Peter Brown）。就连当年资历最浅的小字辈人物拉法特（John Laffety）现在都成了了不起的学者。

上个世纪 70 年代的 IBM 有点像上个世纪 90 年代的微软和过去 10 年（施密特时代）的 Google, 给予杰出科学家做任何有兴趣研究的自由。在那种宽松的环境里，贾里尼克等人提出了统计语音识别的框架结构。在贾里尼克之前，科学家们把语音识别问题当作人工智能和模式匹配问题。而贾里尼克把它当成通信问题，并用两个隐含马尔可夫模型（声学模型和语言模型）把语音识别概括得清清楚楚。这个框架结构对至今的语音和语言处理有着深远的影响，它不仅从根本上使得语音识别有实用的可能，而且奠定了今天自然语言处理的基础。贾里尼克本人后来也因此当选美国工程院院士，并且被 *Technology* 杂志评为 20 世纪 100 名发明家之一。

贾里尼克的前辈香农等人在将统计的方法用于自然语言处理时，遇到了两个不可逾越的障碍：缺乏计算能力强大的计算机和大量可以用于统计的机读文本语料。最后，他的前辈们不得不选择放弃。在上个世纪 70 年代的 IBM，虽然计算机的计算能力不能和今天相比，但是，已经可以做不少事情了。贾里尼克和他的同事需要解决的问题就是如何去找到大量的机读语料。这在今天已经不是问题的问题，在当时可是有点麻烦，因为当时不仅没有网页，连出版物大多都没有电子版的记录，即使有，也在不同的出版商手里，很难收集全。好在当时有一项全球性的业务是通过全球电信网连接在一起的，就是电传。IBM 的科学家最初就是通过电传业务的文本开始进行自然语言处理研究的。

回想起来，基于统计的自然语言处理方法由在上个世纪 70 年代的 IBM

奠定，有着历史的必然性。首先，只有IBM有足够强大的计算功能和数据。其次，贾里尼克（等人）已经在这个领域做了十多年的理论研究，且当时正在IBM工作。第三，上个世纪70年代是小沃森将IBM的业务发展到顶点的时代，IBM对基础研究的投入强度非常大。如果当时的年轻人能看到这几点，又有足够好的数学基础（这是当时贾里尼克等人挑选科学家的必要条件），应该加入IBM，这样一定是前途无量。

贾里尼克和波尔、库克以及拉维夫对人类的另一大贡献是BCJR算法，这是今天数字通信中应用最广的两个算法之一（另一个是维特比算法）。有趣的是，这个算法发明了20年后，才得以广泛应用。IBM于是把它列为IBM有史以来对人类的最大贡献之一，并贴在加州阿莫顿实验室（Amaden Research Labs）墙上。遗憾的是BCJR四个人已经全部离开IBM，有一次IBM的通信部门需要用这个算法，还得从斯坦福大学请一位专家去讲解，这位专家看到IBM橱窗里的成就榜，感慨万分。

1999年在美国凤凰城召开的ICASSP年会上，贾里尼克以“从水门事件到莫妮卡·莱温斯基”为题做了大会报告，总结了语音识别领域30年的成就。重点回顾了当年IBM的工作，以及后来约翰·霍普金斯大学的工作，也包括我的工作。

很多年后我和阿尔弗雷德·斯伯格特（Alfred Spector）[7]谈论为什么当初是没有什么语音识别基础的IBM而不是在这个领域有很长研究时间的AT&T贝尔实验室或者卡内基-梅隆大学提出统计语音识别和语言处理。斯伯格特认为这是因为没有基础的IBM反而不受条条框框的束缚。这是一个方面，而我强调的则是，大多数时候，很多的历史偶然性背后有着它必然的原因，统计自然语言处理诞生于IBM看似有些偶然，但是当时只有IBM有这样的计算能力，又有物质条件同时聚集起一大批世界上最聪明的头脑。

7 先后担任IBM和Google主管研究的副总裁。

3 一位老人的奇迹

读过《浪潮之巅》的读者可能还记得，上个世纪 80 年代末到 90 年代初，是 IBM 最艰难的时期，也是郭士纳大量削减科研经费的时期。不幸的是，语音识别和自然语言处理的研究也在郭士纳削减的名单里。贾里尼克和 IBM 一批最杰出的科学家在上个世纪 90 年代初离开了 IBM，他们中的大多数在华尔街取得了巨大的成功，每个人都成为了千万（可能有的是亿万）富翁。贾里尼克已经到了退休的年龄，他的财富足以让他舒舒服服地安度晚年。但他是一个一辈子都闲不下来的人，而且书生气很浓，于是 1994 年去约翰·霍普金斯大学建立了世界著名的 CLSP（Center for Language and Speech Processsing）实验室。

在贾里尼克到约翰·霍普金斯大学以前，这所以医学院闻名于世的大学在工程领域学科趋于老化，早已经没有了二战前可以和麻省理工学院或者加州理工学院比肩的可能，也完全没有语音识别和自然语言处理这样的新兴学科。贾里尼克从头开始，在短短两三年内就将 CLSP 变成世界一流的研究中心。他主要做了两件大事，两件小事。两件大事是，首先，从美国政府主管研究的部门那里申请到了很多研究经费，然后，每年夏天，他用一部分经费，邀请世界上 20-30 名顶级的科学家和学生到 CLSP 一起工作，使得 CLSP 成为世界上语音和语言处理的中心之一。两件小事是，首先，他招募了一批当时很有潜力的年轻学者，比如今天在自然语言处理方面颇负盛名的雅让斯基和今天 eBay 的 CTO 布莱尔。第二，他利用自己的影响力，在暑期把他的学生派到世界上最好的公司去实习，通过这些学生的优异表现，树立起 CLSP 在培养人才方面的声誉。10 多年后，由于国家安全的需要，美国政府决定在一所一流大学里建立一个信息处理的国家级研究中心（Center of Excellence），贾里尼克领导的约翰·霍普金斯大学的科学家们，在竞标中击败他们在学术界的老对手麻省理工学院和卡内基 - 梅隆大学，将这个中心落户到约翰·霍普金斯大学，确立了他在这个学术领域的世界级领导地位。

贾里尼克治学极为严谨，对学生要求也极严。他淘汰学生的比例极高，即使留下来的，毕业时间也极长。但是，另一方面，贾里尼克也千方百计利用自己的影响力为学生的学习和事业提供便利。贾里尼克为组里的每一位学生提供从进组第一天到离开组之前最后一天全部的学费和生活费。他还为每一位学生联系实习机会，并保证每位学生在博士生阶段至少在大公司实习一次。从他那里拿到博士学位的学生，全部任职于著名实验室，比如 IBM、微软、AT&T 和 Google 的实验室。为了提高外籍学生的英语水平，贾里尼克用自己的经费为他们请私人英语教师。

贾里尼克教授桃李满天下，这里面包括他的学生、过去的下属以及在学术界众多沿袭他的研究方法的晚辈，比如 Google 研究院的院长诺威格（Peter Norvig）和费尔南多·皮耶尔（Fernando Pereira），这些人分布在世界上主要的大学和公司的研究所，渐渐地形成了一个学派。而贾里尼克是这个学派的精神领袖。

贾里尼克教授在学术上给我最大的帮助就是提高了我在学术上的境界。他告诉我最多的是：什么方法不好。在这一点上他与股神巴菲特给和他吃饭的投资人[8]的建议有异曲同工之处。巴菲特和那些投资人讲，你们都非常聪明，不需要我告诉你们做什么，我只需要告诉你们不要去做什么（这样可以少犯很多错误），这些不要做的事情，是巴菲特从一生的经验教训中得到的。贾里尼克会在第一时间告诉我什么方法不好，因为在 IBM 时他和他的同事吃过这方面的亏。至于什么方法好，他相信我比他强，自己能找到。所以他节省了我很多可能做无用功的时间。同时，他考虑问题的方法让我终身受益。

8 巴菲特每年和一位投资人共进晚餐，取决于哪位投资人出价高。这个出价由巴菲特捐给慈善机构。

贾里尼克生活俭朴，一辆老式丰田车开了 20 多年，比组里学生的车都破。他每年都邀请组里的学生和教授到家里做客，很多已毕业的学生也专程赶来聚会。在那里，他不再谈论学术问题，而会谈些巩俐的电影（他太太是哥伦比亚大学电影专业的教授），或是一些科学家的八卦，比如著名的信息论专家、斯坦福大学的科弗（Thomas Cover）教授如何被拉斯

韦加斯的赌馆定为不受欢迎的人，等等。但是他家里聚会上的食物实在难吃，无非是些生胡萝卜和芹菜。后来贾里尼克掏钱让系里另一个教授米勒承办聚会，米勒教授每次都请专业大厨在家做出极丰盛的晚宴，并准备许多美酒，从此这种聚会就转移到米勒家了。

贾里尼克的太太米兰娜是哥伦比亚大学电影领域的教授，可能是受他太太影响，他很早就开始观看中国电影。中国早期走向世界的电影，女主角基本上都是巩俐，所以他很奇怪为什么这么大的国家只有这么一位女演员。此外，贾里尼克早期对中国的了解就是清华大学和青岛啤酒了。他多次把这两个名字搞混，有两次被香港科技大学的冯雁（Pascale Fung）教授抓住这个错误。

贾里尼克说话心直口快，不留余地。在他面前谈论学术一定要十分严谨，否则很容易被他抓住辫子。除了刚才提到的对语言学家略有偏见的评论，他对许多世界级的大师都有过很多“刻薄”但又实事求是的评论，这些评论在业界广为流传。当然，当一个人真正做出成绩时，贾里尼克还是毫不吝惜他的赞美之词的。1999 年，我在欧洲语言大会 Eurospeech 上获得了最佳论文奖，贾里尼克在实验室里一见到我就讲“我们以你为荣”（We are proud of you.），并且后来多次提及此事。贾里尼克在 40 多年的学术生涯中居然没有得罪太多人，可以说是一个奇迹。我想这除了他的成就之外，还因为他是一个公正的人。

前面讲过，贾里尼克是一个闲不住的人。我经常看到他周末到实验室加班。他在 70 多岁以后依然头脑敏锐，并且每天按时上班。2010 年 9 月 14 日，他像往常一样来到办公室，但不幸的是，因为心脏病发作在办公桌前过世了。我听到这个消息时又悲伤又震惊，因为几个月前我去约翰·霍普金斯大学看他时，他还是好好的。他在别人退休、安度晚年的年龄开始创立当今世界学术界最大的语音和语言处理中心，并且工作到了生命的最后一天。很多年前他和我谈论学习是一辈子的事情，他确实做到了。

由于他有大量的学生和朋友在 Google 工作，这些人和 Google 公司为约翰·霍普金斯大学捐赠了一笔钱，用于创立贾里尼克奖学金。有志从事这个领域研究的大学生，可以去申请这个奖学金[9]。

9 http://eng.jhu.edu/wse/Fred_Jelinek/memory/jelinek-fellowship

第8章 简单之美——布尔代数和搜索引擎的索引

在接下来的几章里，我们会介绍与搜索有关的技术。几年前，当这个系列在谷歌黑板报上登出时，很多读者都很想通过它知道Google的独门搜索技术，对我只讲述简单的原理很失望。这次，我可能还是要让一些读者失望了，因为我依然不会讲得很深。主要有这样几个原因，首先我希望这本书的读者是大众，而不仅仅是搜索引擎公司的工程师。对于前者，帮助他们了解数学在工程中的作用，远比了解与他们的工作无关的算法要有意义得多。第二，技术分为术和道两种，具体的做事方法是术，做事的原理和原则是道。这本书的目的是讲道而不是术。很多具体的搜索技术很快会从独门绝技到普及，再到落伍，追求术的人一辈子工作很辛苦。只有掌握了搜索的本质和精髓才能永远游刃有余。第三，很多希望我介绍“术”的人是希望走捷径。但是真正做好一件事没有捷径，需要一万小时的专业训练和努力。做好搜索，最基本的要求是每天分析10-20个不好的搜索结果，累积一段时间才有感觉。我在Google做搜索质量的时候每天分析的搜索数量远不止这个，Google的搜索质量第一技术负责人阿米特·辛格（Amit Singhal）今天依然经常分析这些不好的结果。但是，很多做搜索的工程师（美国的、中国的都有）做不到这一点，总是希望一个算法、一个模型就能毕其功于一役，这是不现实的。

现在我们回到搜索引擎这个话题。搜索引擎的原理其实非常简单，建立一个搜索引擎大致需要做这样几件事：自动下载尽可能多的网页；建立快速有效的索引；根据相关性对网页进行公平准确的排序。所以我到腾讯以后，就把搜搜所有的搜索产品提炼成下载、索引和排序这三种基本服务。这就是搜索的“道”。所有的搜索服务都可以在这三个基本服务的基础上很快实现，这就是搜索的“术”。

在腾讯内部升级搜索引擎时，首先要改进和统一的就是所有搜索业务的索引，否则提高搜索质量就如同浮沙建塔一样不稳固。同样我们在这本书中介绍搜索，也从索引出发，因为它最基础，也最重要。

1 布尔代数

世界上不可能有比二进制更简单的计数方法了，它只有两个数字：0 和 1。从单纯数学的角度讲，它甚至比我们的十进制更合理。但是我们人有十个手指，使用起来比二进制（或者八进制）方便得多，所以在进化和文明发展过程中人类采用了十进制。二进制的历史其实也很早，中国古代的阴阳学说可以认为是最早二进制的雏形。而二进制作为一个计数系统，公元前 2-5 世纪时由印度学者完成，但是他们没有使用 0 和 1 计数。到 17 世纪，德国伟大的数学家莱布尼兹（Gottfried Leibniz）把它完善，并且用 0 和 1 表示它的两个数字，成为我们今天使用的二进制。二进制除了是一种计数的方式外，它还可以表示逻辑的“是”与“非”。这第二个特性在索引中非常有用。布尔运算是针对二进制，尤其是二进制第二个特性的运算，它很简单，可能没有比布尔运算更简单的运算了。尽管今天每个搜索引擎都宣称自己如何聪明、多么智能（这个词非常忽悠人），其实从根本上讲都没有逃出布尔运算的框框。

布尔（George Boole）是 19 世纪英国的一位中学数学老师，还创办过一所中学。后来在爱尔兰科克（Cork）的一所学院当教授。生前没有人认为他是数学家，虽然他曾经在剑桥大学数学杂志（*Cambridge*

Mathematical Journal）上发表过论文。（英国另一位生前没有被公认为科学家的是著名物理学家焦耳，虽然他生前已经是英国皇家科学院院士，但是他的公认身份是啤酒商。）布尔在工作之余，喜欢阅读数学论著，思考数学问题。1854年，布尔的《思维规律》（*An Investigation of the Laws of Thought, on which are founded the Mathematical Theories of Logic and Probabilities*）一书，第一次向人们展示了如何用数学的方法解决逻辑问题。在此之前，人们普遍的认识是数学和逻辑是两个不同的学科，今天联合国教科文组织依然把它们严格分开。

布尔代数简单得不能再简单了。运算的元素只有两个：1（TRUE，真）和0（FALSE，假）。基本的运算只有"与"（AND）、"或"（OR）和"非"（NOT）三种（后来发现，这三种运算都可以转换成"与非"AND-NOT一种运算）。全部运算只用下列几张真值表就能完全描述清楚。

表8.1 与运算真值表

AND	1	0
1	1	0
0	0	0

表8.1说明，如果AND运算的两个元素有一个是0，则运算结果总是0。如果两个元素都是1，运算结果是1。例如，"太阳从西边升起"这个判断是假的（0），"水可以流动"这个判断是真的（1），那么，"太阳从西边升起并且水可以流动"就是假的（0）。

表8.2 或运算真值表

OR	1	0
1	1	1
0	1	0

表8.2说明，如果OR运算的两个元素有一个是1，则运算结果总是1。

如果两个元素都是 0，则运算结果是 0。比如说，“张三是比赛第一名”这个结论是假的（0），“李四是比赛第一名”是真的（1），那么“张三或者李四是第一名”就是真的（1）。

表 8.3 非运算真值表

NOT	
1	0
0	1

表 8.3 说明，NOT 运算把 1 变成 0，把 0 变成 1。比如，如果“象牙是白的”是真的（1），那么“象牙不是白的”必定是假的（0）。

读者也许会问这么简单的理论能解决什么实际问题。和布尔同时代的数学家们也有同样的疑问。事实上，在布尔代数提出后 80 多年里，它确实没有什么像样的应用，直到 1938 年香农在他的硕士论文中指出用布尔代数来实现开关电路，才使得布尔代数成为数字电路的基础。所有的数学和逻辑运算，加、减、乘、除、乘方、开方等等，全都能转换成二值的布尔运算。我们在第一章讲到，数学的发展实际上是不断地抽象和概括的过程，这些抽象了的方法看似离生活越来越远，但是它们最终能找到适用的地方，布尔代数便是如此。

现在看看文献检索和布尔运算的关系。对于一个用户输入的关键词，搜索引擎要判断每篇文献是否含有这个关键词，如果一篇文献含有它，我们相应地给这篇文献一个逻辑值 —— 真（TRUE 或 1），否则，给一个逻辑值 —— 假（FALSE 或 0）。比如要找有关原子能应用的文献，但并不想知道如何造原子弹。可以这样写一个查询语句“原子能 AND 应用 AND (NOT 原子弹)”，表示符合要求的文献必须同时满足三个条件：

包含原子能，包含应用，不包含原子弹

一篇文献对于上面每一个条件，都有一个 TRUE 或者 FALSE 的答案。根据上述真值表就能算出每篇文献是否是要找的。这样逻辑推理和计算就合二为一了。

布尔代数对于数学的意义等同于量子力学对于物理学的意义，它们将我们对世界的认识从连续状态扩展到离散状态。在布尔代数的“世界”里，万物都是可以量子化的，从连续的变成一个个的，它们的运算“与、或、非”也就和传统的代数运算完全不同了。现代物理的研究成果表明，我们的世界实实在在是量子化的而不是连续的。我们的宇宙的基本粒子数目是有限的[1]，而且远比古高尔（10^{100}）要小得多。

1 据 http://www.universetoday.com/36302/atoms-in-the-universe/ 估计宇宙中原子的数量是 10^{78}-10^{82}，如果按照最小的基本粒子（夸克、电子、光子等）统计，再考虑到暗物质和暗能量，折算下来不应该超过 10^{86}。

2 索引

大部分使用搜索引擎的人都会吃惊为什么它能在零点零几秒钟找到成千上万甚至上亿的搜索结果。显然，如果是扫描所有的文本，计算机扫描的速度再快也不可能做到这一点，这里面一定暗藏技巧。这个技巧就是建索引。这就如同我们科技读物背后的索引，或者图书馆的索引。Google 有一道面试产品经理的考题，就是“如何向你的奶奶解释搜索引擎”。大部分候选人都是试图从互联网、搜索等等产品的技术层面给出解释，这道题基本通不过。好的回答是拿图书馆的索引卡片做类比。每个网站就像图书馆里的一本书，我们不可能在图书馆书架上一本本地找，而是要通过搜索卡片找到它的位置，然后直接去书架上拿。

图书馆的索引卡片当然无法进行复杂的逻辑运算。但是，当信息检索进入计算机时代后，图书的索引便不再是卡片，而是基于数据库的。数据库的查询语句（SQL）支持各种复杂的逻辑组合，但是背后的基本原理是基于布尔运算的，至今如此。早期的文献检索查询系统，严格要求查询语句符合布尔运算。相比之下，今天的搜索引擎要聪明得多，它会自动把用户的查询语句转换成布尔运算的算式。但是基本的原理没有什么不同。

最简单的索引的结构是用一个很长的二进制数表示一个关键字是否出现在每篇文献中。有多少篇文献，就有多少位数，每一位对应一篇文献，1 代表相应的文献有这个关键字，0 代表没有。比如关键字“原子能”对应

的二进制数是 0100100011000001...，表示第二、第五、第九、第十、第十六篇文献包含这个关键字。上述过程其实就是将一篇篇千差万别的文本进行量子化的过程。注意，这个二进制数非常之长。同样，假定“应用”对应的二进制数是 0010100110000001...，那么要找到同时包含“原子能”和“应用”的文献时，只要将这两个二进制数进行布尔运算 AND。根据上面的真值表，我们知道运算结果是 0000100000000001...，表示第五篇、第十六篇文献满足要求。

注意，计算机做布尔运算是非常非常快的。现在最便宜的微机都可以在一个指令周期进行 32 位布尔运算，一秒钟进行数十亿次以上。当然，由于这些二进制数中的绝大部分位数都是零，只需要记录那些等于 1 的位数即可。于是，搜索引擎的索引就变成了一张大表：表的每一行对应一个关键词，而每一个关键词后面跟着一组数字，是包含该关键词的文献序号。

对于互联网的搜索引擎来讲，每一个网页就是一个文献。互联网的网页数量是巨大的，网络中所用的词也非常非常多。因此，这个索引是巨大的，在万亿字节这个量级。早期的搜索引擎（比如 AltaVista 以前的所有搜索引擎），由于受计算机速度和容量的限制，只能对重要的关键的主题词建立索引。至今很多学术杂志还要求作者提供 3-5 个关键词。这样所有不常见的词和太常见的虚词就找不到了。现在，为了保证对任何搜索都能提供相关的网页，主要的搜索引擎都是对所有的词进行索引。但是，这在工程上却是一件很有挑战性的事情。

假如互联网上有 100 亿[2]（10^{10}）个有意义的网页，而词汇表的大小是 30 万（也是保守估计的数字），那么这个索引的大小至少是 100 亿 ×30 万＝3 000 万亿。考虑到大多数词只出现在一部分文本中，压缩比为 100:1，也是 30 万亿的量级。为了网页排名方便，索引中还需存有大量附加信息，诸如每个词出现的位置、次数等等。因此，整个索引就变得非常之大，显然，这不是一台服务器的内存能够存下的。所以，这些索引需要通过分布式

2 实际数量比这个多。

的方式存储到不同的服务器上。普遍的做法就是根据网页的序号将索引分成很多份(Shards),分别存储在不同的服务器中。每当接受一个查询时,这个查询就被分发到许许多多服务器中，这些服务器同时并行处理用户请求，并把结果送到主服务器进行合并处理，最后将结果返回给用户。

随着互联网上内容的增加，尤其是互联网 2.0 时代，用户产生的内容越来越多，即使是 Google 这样的服务器数量近乎无限的公司，也感到了数据增加的压力。因此，根据需要网页的重要性、质量和访问的频率建立常用和非常用等不同级别的索引。常用的索引需要访问速度快，附加的信息多，更新也要快；而非常用的要求就低多了。但是不论搜索引擎的索引在工程上如何复杂，原理上依然非常简单，即等价于布尔运算。

3　小结

布尔代数非常简单，但是对数学和计算机发展的意义重大，它不仅把逻辑和数学合二为一，而且给了我们一个全新的视角看待世界，开创了今天数字化的时代。在此，让我们用伟大的科学家牛顿的一句话来结束这一章，“(人们)发觉真理在形式上从来是简单的，而不是复杂和含混的。”(Truth is ever to be found in simplicity, and not in the multiplicity and confusion of things.)

第 9 章　图论和网络爬虫

离散数学是当代数学的一个重要分支，也是计算机科学的数学基础。它包括数理逻辑、集合论、图论和近世代数四个分支。数理逻辑基于布尔运算，前面已经介绍过了。这里介绍图论和互联网自动下载工具网络爬虫 之间的关系。顺便提一句，用 Google Trends 来搜索一下“离散数学”这个词，可以发现不少有趣的现象。比如，武汉、哈尔滨、合肥和长沙这些城市对这一数学主题最有兴趣。

第 8 章谈到了如何建立搜索引擎的索引，那么如何自动下载互联网所有的网页呢？这需要用到图论中的遍历（Traverse）算法。

1　图论

图论的起源可追溯到大数学家欧拉（Leonhard Euler）诞生的那个年代。1736 年，欧拉来到普鲁士的哥尼斯堡（Konigsberg，大哲学家康德的故乡，现在是俄罗斯的加里宁格勒），发现当地居民有一项消遣活动，就是试图将下图中的每座桥恰好走过一遍并回到原出发点，从来没有人成功过。欧拉证明了这种走法是不可能的，并写了一篇论文，一般认为这是图论的开始。至于为什么不可能，我们在延伸阅读里会介绍。

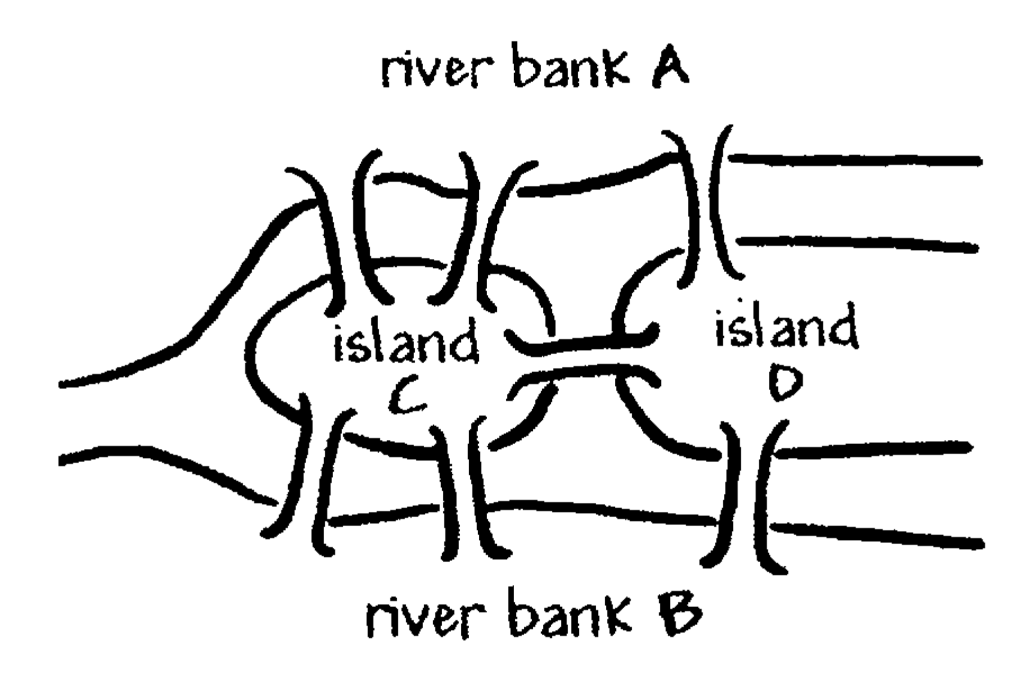


图 9.1 哥尼斯堡的七座桥

图论中所讨论的图由一些节点和连接这些节点的弧组成。如果我们把中国的城市当成节点，连接城市的国道当成弧，那么全国的公路干线网就是图论中所说的图。关于图的算法有很多，但最重要的是图的遍历算法，也就是如何通过弧访问图的各个节点。以中国公路网为例，我们从北京出发，访问所有的城市。可以先看一看北京和哪些城市直接相连，比如说和天津、济南、石家庄、沈阳、呼和浩特直接相连（图 9.2 中的黑色线条）。当然，这些城市之间还可以有其他的连接（图 9.2 中的灰色线）。

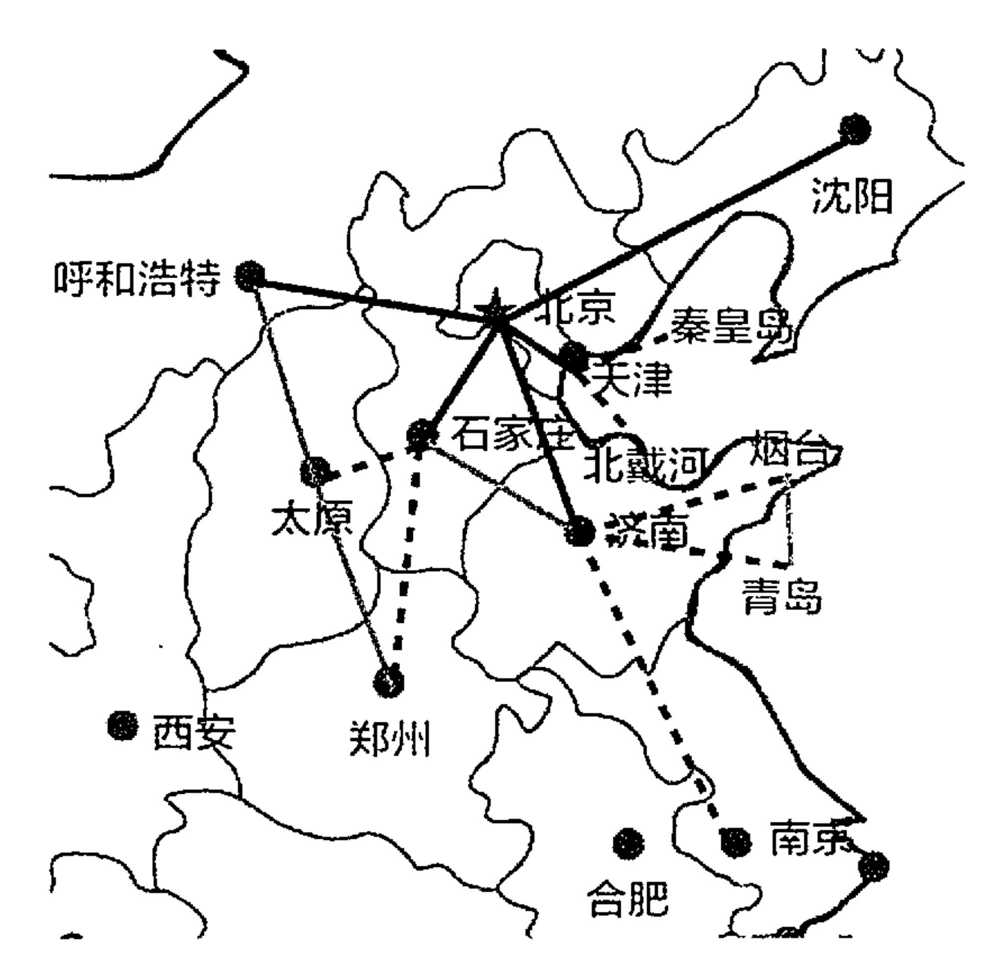


图 9.2 中国公路图

从北京出发，可以依次访问这些城市。先访问那些直接和北京相连的城市，比如天津、济南等。然后看看都有哪些城市和这些已经访问过的城市相连，比如说北戴河、秦皇岛与天津相连，青岛、烟台、南京和济南相连，太原、郑州和石家庄相连等（图 9.2 中的虚线），而后再一次访问北戴河这些城市，直到把中国所有的城市都访问过一遍为止。这种图的遍历算法称为“广度优先搜索”（Breadth-First Search，简称 BFS），因为它先要尽可能“广”地访问每个节点所直接连接的其他节点。见图 9.3：

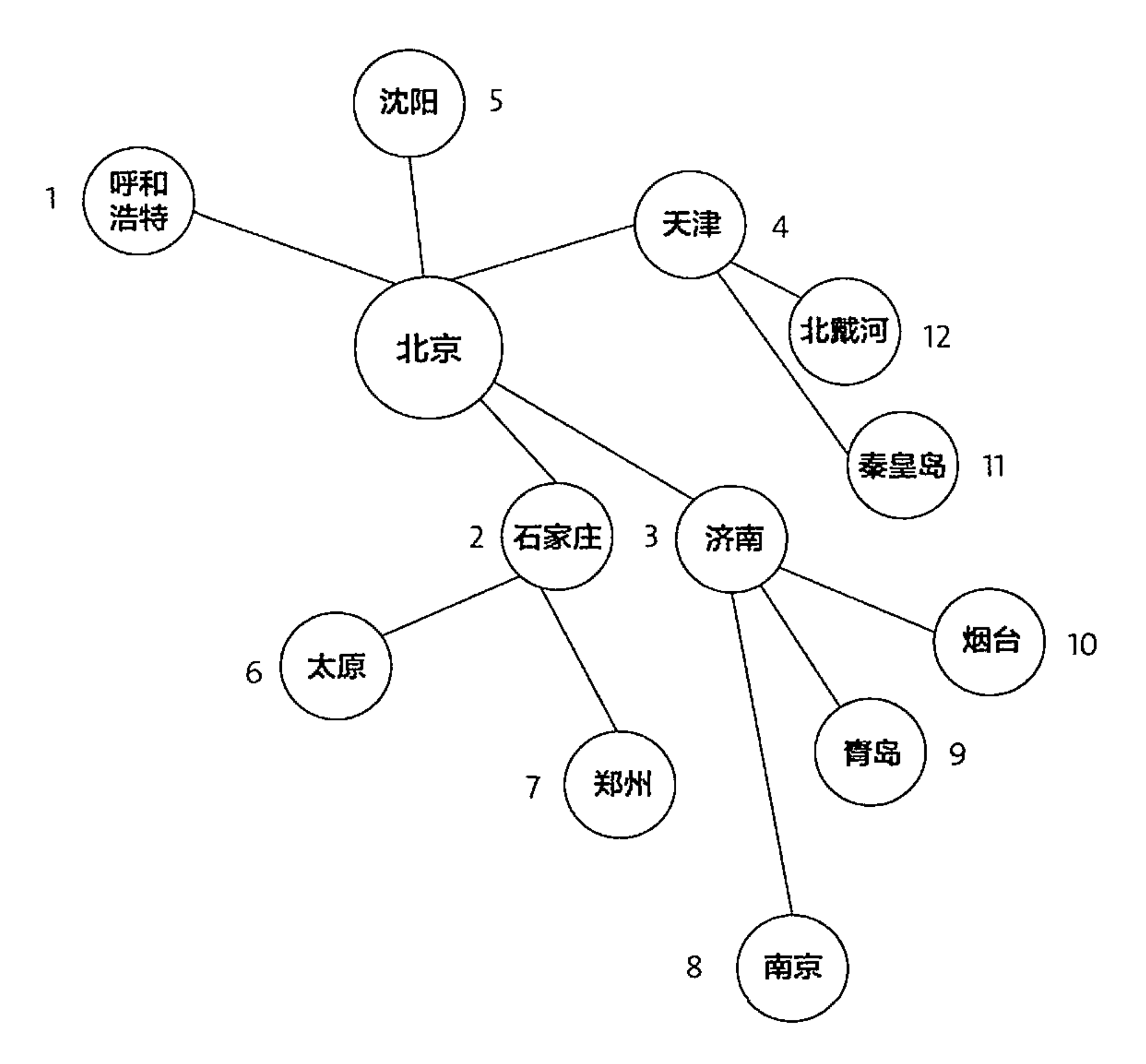


图 9.3　广度优先遍历，图中的数字表示遍历的次序

另外还有一种策略是从北京出发，随便找一个相连的城市，作为到下一个要访问的城市，比如说济南，然后从济南出发到下一个城市，比如说南京，再访问从南京出发的城市，一直走到头，直到找不到更远的城市了，再往回找，看看中间是否有尚未访问的城市。这种方法叫“深度优先搜索”（Depth-First Search，简称 DFS），因为它是一条路走到黑。下图是采用深度优先搜索算法时遍历整个图的次序。

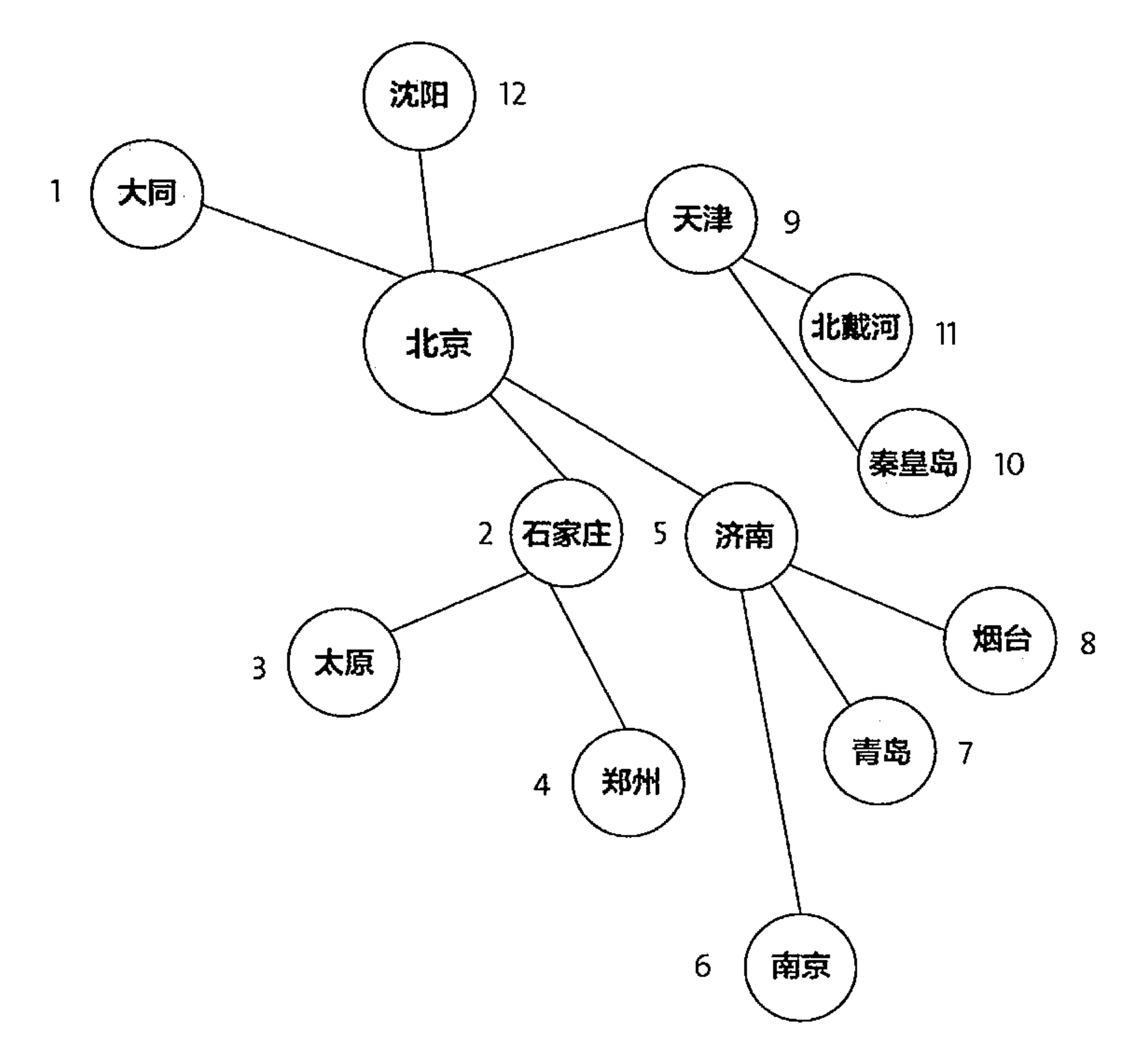


图 9.4　深度优先遍历，各个城市的访问次序

这两种方法都可以保证访问到全部的城市。当然，不论采用哪种方法，都应该用一个小本本记录已经访问过的城市，避免同一个城市访问多次或者漏掉哪个城市。

2　网络爬虫

现在看看图论的遍历算法和搜索引擎的关系。互联网虽然很复杂，但是说穿了其实就是一张大图而已 —— 可以把每一个网页当作一个节点，把那些超链接（Hyperlinks）当作连接网页的弧。很多读者可能已经注意到，网页中那些蓝色、带有下划线的文字背后其实藏着对应的网址，当你点击的时候，浏览器通过这些隐含的网址跳转到相应的网页。这些隐含在文字背后的网址称为“超链接”。有了超链接，我们可以从任何一个网页出发，用图的遍历算法，自动地访问到每一个网页并把它们存起来。完成这个功能的程序叫做网络爬虫（Web Crawlers），或者在一些文献

中称为“机器人”（Robot)。世界上第一个网络爬虫是由麻省理工学院的学生马休·格雷（Matthew Gray）在1993年写成的。他给自己的程序起了个名字叫“互联网漫游者”（WWW Wanderer）。以后的网络爬虫越写越复杂，但原理是一样的。

我们来看看网络爬虫如何下载整个互联网。假定从一家门户网站的首页出发，先下载这个网页，然后通过分析这个网页，可以找到页面里的所有超链接，也就等于知道了这家门户网站首页所直接链接的全部网页，诸如雅虎邮件、雅虎财经、雅虎新闻等。接下来访问、下载并分析这家门户网站的邮件等网页，又能找到其他相连的网页。让计算机不停地做下去，就能下载整个的互联网。当然，也要记载哪个网页下载过了，以免重复。在网络爬虫中，使用一个称为“哈希表”（Hash Table）的列表而不是一个记事本记录网页是否下载过的信息。

现在的互联网非常庞大，不可能通过一台或几台计算机服务器就能完成下载任务。比如Google在2010年时整个的索引大小大约有5 000亿个网页，即使更新最频繁的基础索引也有100亿个网页，假如下载一个网页需要一秒钟，下载这100亿个网页则需要317年，如果下载5 000亿个网页则需要16 000年左右，是我们人类有文字记载历史的三倍时间。因此，一个商业的网络爬虫需要有成千上万个服务器，并且通过高速网络连接起来。如何建立起这样复杂的网络系统，如何协调这些服务器的任务，就是网络设计和程序设计的艺术了。

3 延伸阅读：图论的两点补充说明

3.1 欧拉七桥问题的证明

把每一块连通的陆地作为一个顶点，每一座桥当成图的一条边，那么就把哥尼斯堡的七座桥抽象成下面的图。

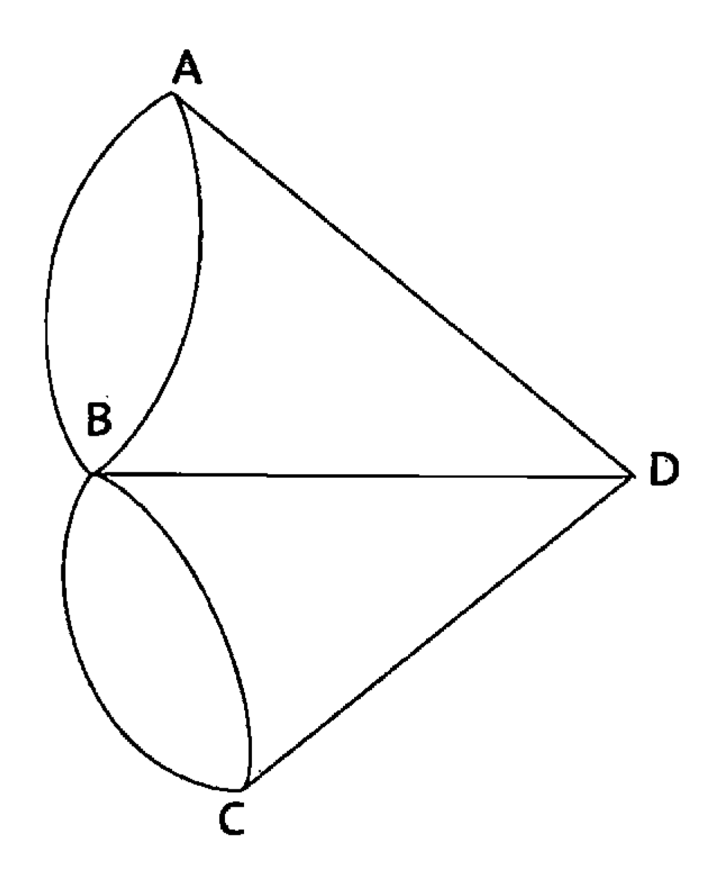


图 9.5　哥尼斯堡七桥的抽象图

对于图中的每一个顶点，它相连的边的数量定义为它的度（Degree）。

定理：如果一个图能够从一个顶点出发，每条边不重复地遍历一遍回到这个顶点，那么每一顶点的度必须为偶数。

证明：假如能够遍历图的每一条边各一次，那么对于每个顶点，需要从某条边进入顶点，同时从另一条边离开这个顶点。进入和离开顶点的次数是相同的，因此每个顶点有多少条进入的边，就有多少条出去的边。也就是说，每个顶点相连的边的数量是成对出现的，即每个顶点的度都是偶数。

在图 9.5 中，有多个顶点的度为奇数，因此，这个图无法从一个顶点出发，遍历每条边各一次然后回到这个顶点。

3.2　构建网络爬虫的工程要点

“如何构建一个网络爬虫”是我在 Google 最常使用的一道面试题。因为我经常使用，一些面试者其实知道这个事实。但是我依然使用，而且依然可以有效地考察出一个候选人的计算机科学理论基础、算法能力和他的工程素养。这道题漂亮的地方在于它没有完全对和错的答案，但是有

好和不好、可行和不可行的答案，而且可以不断地往深处问下去。一个好的候选人不需要做过网络爬虫也能很好回答这道题，而那些仅仅有执行能力的三流工程师，即使在做网络爬虫的工作，里面很多地方也不会考虑全面。

网络爬虫在工程实现上要考虑的细节非常多，其中大的方面有这样几点：

首先，用 BFS 还是 DFS？

虽然从理论上讲，这两个算法（在不考虑时间因素的前提下）都能够在大致相同的时间[1]里“爬下”整个“静态”互联网上的内容，但是工程上的两个假设 —— 不考虑时间因素、互联网静态不变，都是现实中做不到的。搜索引擎的网络爬虫问题更应该定义成“如何在有限时间里最多地爬下最重要的网页”。显然各个网站最重要的网页应该是它的首页。在最极端的情况下，如果爬虫非常小，只能下载非常有限的网页，那么应该下载的是所有网站的首页，如果把爬虫再扩大些，应该爬下从首页直接链接的网页（就如同和北京直接相连的城市），因为这些网页是网站设计者自己认为相当重要的网页。在这个前提下，显然 BFS 明显优于 DFS。事实上在搜索引擎的爬虫里，虽然不是简单地采用 BFS，但是先爬哪个网页，后爬哪个网页的调度程序，原理上基本上是 BFS。

1 是节点数量V和边的数量E之和的线性函数，即$O(V+E)$。

那么是否 DFS 就不使用了呢？也不是这样的。这是和爬虫的分布式结构以及网络通信的握手成本有关。所谓“握手”就是指下载服务器和网站的服务器建立通信的过程。这个过程需要额外的时间（Overhead Time），如果握手的次数太多，下载的效率就降低了。实际的网络爬虫都是一个由成百上千甚至成千上万台服务器组成的分布式系统。对于某个网站，一般是由特定的一台或者几台服务器专门下载。这些服务器下载完一个网站，然后再进入下一个网站，而不是每个网站先轮流下载 5%，然后再回过头来下载第二批。这样可以避免握手的次数太多。如果是下载完第一个网站再下载第二个，那么这又有点像 DFS，虽然下载同一个网站（或者子网站）时，还是需要用 BFS 的。

总结起来，网络爬虫对网页遍历的次序不是简单的BFS或者DFS，而是有一个相对复杂的下载优先级排序的方法。管理这个优先级排序的子系统一般称为调度系统（Scheduler）。由它来决定当一个网页下载完成后，接下来下载哪一个。当然在调度系统里需要存储那些已经发现但是尚未下载的网页的URL，它们一般存在一个优先级队列（Priority Queue）里。而用这种方式遍历整个互联网，在工程上和BFS更相似。因此，在爬虫中，BFS的成分多一些。

第二，页面的分析和URL的提取。

在上一节中提到，当一个网页下载完成后，需要从这个网页中提取其中的URL，把它们加入到下载的队列中。这个工作在互联网的早期不难，因为那时的网页都是直接用HTML语言书写的。那些URL都以文本的形式放在网页中，前后都有明显的标识，很容易提取出来。但是现在很多URL的提取就不那么直接了，因为很多网页如今是用一些脚本语言（比如JavaScript）生成的。打开网页的源代码，URL不是直接可见的文本，而是运行这一段脚本后才能得到的结果。因此，网络爬虫的页面分析就变得复杂很多，它要模拟浏览器运行一个网页，才能得到里面隐含的URL。有些网页的脚本写得非常不规范，以至于解析起来非常困难。可是，这些网页还是可以在浏览器中打开，说明浏览器可以解析。因此，需要做浏览器内核的工程师来写网络爬虫中的解析程序，可惜出色的浏览器内核工程师在全世界数量并不多。因此，若你发现一些网页明明存在，但搜索引擎就是没有收录，一个可能的原因是网络爬虫中的解析程序没能成功解析网页中不规范的脚本程序。

第三，记录哪些网页已经下载过的小本本——URL表。

在互联网上，一个网页可能被多个网页中的超链接所指向，即在互联网这张大图上，有很多弧（链接）可以走到这个节点（网页）。这样在遍历互联网这张图的时候，这个网页可能被多次访问到。为了防止一个网页被下载多次，需要在一个哈希表中记录哪些网页已经下载过。再遇到

这个网页时，我们就可以跳过它。采用哈希表的好处是，判断一个网页的 URL 是否在表中，平均只需要一次（或者略多的）查找。当然，如果遇到没有下载的网页，除了下载该网页，还需要在下载完成后，将这个网页的 URL 存到哈希表中，这个操作对哈希表来讲也非常简单。在一台下载服务器上建立和维护一张哈希表并不是难事。但是如果同时有上千台服务器一起下载网页，维护一张统一的哈希表就不是一件容易的事情了。首先，这张哈希表会大到一台服务器存储不下。其次，由于每个下载服务器在开始下载前和完成下载后都要访问和维护这张表，以免不同的服务器做重复的工作，这个存储哈希表的服务器的通信就成了整个爬虫系统的瓶颈。如何消除这个瓶颈是我经常考应聘者的试题。

这里有各种解决办法，没有绝对正确的，但是却有好坏之分。好的方法一般都采用了这样两个技术：首先明确每台下载服务器的分工，也就是说在调度时一看到某个 URL 就知道要交给哪台服务器去下载，这样就避免了很多服务器对同一个 URL 做出是否需要下载的判断。然后，在明确分工的基础上，判断 URL 是否下载就可以批处理了，比如每次向哈希表（一组独立的服务器）发送一大批询问，或者每次更新一大批哈希表的内容。这样通信的次数就大大减少了。

4 小结

在图论出现后的很长时间里，现实世界中图的大小都是在几千个节点以下的规模（比如公路图、铁路图等）。那时候，图的遍历是一件很简单的事情，因此在工业界没有多少人专门研究这个问题。过去，即使是计算机专业的学生，大部分人也体会不到这个领域的研究有什么实际用处，因为大家在工作中可能一辈子都用不到它。但是随着互联网的出现，图的遍历方法一下子有了用武之地。很多数学方法就是这样，看上去没有什么实际用途，但是随着时间的推移会一下子派上大用场。这恐怕是世界上还有很多人毕生研究数学的原因。

第10章 PageRank — Google的民主表决式网页排名技术

对于大部分用户的查询，今天的搜索引擎，都会返回成千上万条结果，那么应该如何排序，把用户最想看到的结果排在前面呢？这个问题很大程度上决定了搜索引擎的质量。我们在这一章和下一章将回答这个问题。总的来讲，对于一个特定的查询，搜索结果的排名取决于两组信息，关于网页的质量信息(Quality)，和这个查询与每个网页的相关性信息(Relevance)。这一章介绍衡量网页质量的方法，下一章介绍度量搜索关键词和网页相关性的方法。

1 PageRank 算法的原理

大家可能知道，Google革命性的发明是它名为“PageRank”的网页排名算法，这项技术在1998年前后使得搜索的相关性有了质的飞跃，圆满地解决了以往网页搜索结果中排序不好的问题。以至于大家认为Google的搜索质量好，甚至这个公司成功都是基于这个算法。当然，这样的说法有些夸大了。

最先试图给互联网上的众多网站排序的并不是Google，而是雅虎公司。雅虎的创始人杨致远和费罗最早使用目录分类的方式让用户通过互联网检索信息（关于这段历史，读者可以参看拙作《浪潮之巅》）。但由于当时计算机容量和速度的限制，雅虎和同时代的其他搜索引擎都存在一个共

同的问题：收录的网页太少，而且只能对网页中常见内容相关的实际用词进行索引。那时，用户很难找到相关信息。我记得 1999 年以前查找一篇论文，要换好几个搜索引擎。后来 DEC 开发了 AltaVista 搜索引擎，只用了一台 Alpha 服务器，却收录了比以往任何引擎都多的网页，而且对里面的每个词都进行索引。但是，AltaVista 虽然让用户搜索到了大量结果，但大部分结果却与查询不太相关，有时找想看的网页需要翻好几页。所以，最初的 AltaVista 在一定程度上解决了覆盖率的问题，但还不能很好地对结果进行排序。和 AltaVista 同时代的搜索引擎公司还有 Inktomi。这两家公司多少发现互联网网页的质量在搜索结果的排序中也应该起一些作用，于是尝试了一些方法，有点效果，但这些方法都是在数学上不很完善的方法。这些方法或多或少地用到了指向某个网页的链接以及链接上的文本（在搜索技术中称为锚文本，Anchor Text）的技术。这在当时都是公开的技术。1996 年，我在约翰·霍普金斯大学的师兄 Scott Weiss（后来在威廉·玛丽学院任教）在做信息检索博士论文时就用链接数量作为搜索排序的一个因子。



图 10.1 大家都说“他是李开复”

真正找到计算网页自身质量的完美的数学模型的是 Google 的创始人拉里·佩奇和谢尔盖·布林。Google 的“PageRank”（网页排名）是怎么回事呢？其实简单地说就是民主表决。打个比方，假如我们要找李开复博士，有 100 个人举手说自己是李开复。那么谁是真的呢？也许有好几个真的，但即使如此谁又是大家真正想找的呢？如果大家都说在创新工场的那个是真的，那么他就是真的。

在互联网上，如果一个网页被很多其他网页所链接，说明它受到普遍的承认和信赖，那么它的排名就高。这就是 PageRank 的核心思想。当然 Google 的 PageRank 算法实际上要复杂得多。比如说，对来自不同网页的链接区别对待，因为网页排名高的那些网页的链接更可靠，于是要给这些链接以较大的权重。这就好比在现实生活中股东大会里的表决，是要考虑每个股东的表决权（Voting Power）的，拥有 20% 表决权的股东和拥有 1% 表决权的股东，对最后的表决结果的影响力明显不同。PageRank 算法考虑了这个因素，即网页排名高的网站贡献的链接权重大。

现在举一个例子，我们知道一个网页 Y 的排名应该来自于所有指向这个网页的其他网页 $X_1,X_2,\ldots,X_K$ 的权重之和，如下图中，Y 的网页排名 $pagerank = 0.001 + 0.01 + 0.02 + 0.05 = 0.081$。

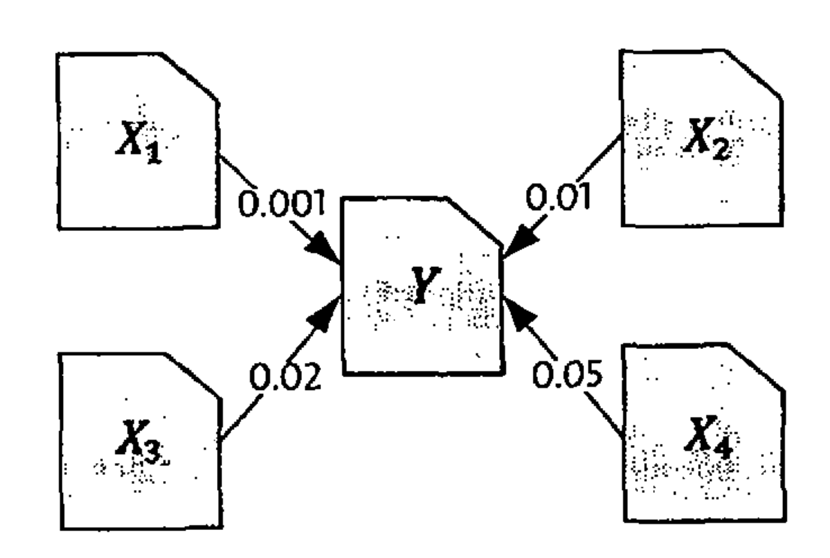


图 10.2 网页排名的计算

虽然佩奇和布林不强调这个算法中谁都贡献了什么思想，但是据我了解，上述想法应该来自于佩奇。接下来的问题是 X_1,X_2,X_3,X_4 的权重分别是多少，如何度量。佩奇认为，应该是这些网页本身的网页排名。现在麻烦来了，计算搜索结果的网页排名过程中需要用到网页本身的排名，这不成了“先有鸡还是先有蛋”的问题了吗？

破解这个怪圈的应该是布林。他把这个问题变成了一个二维矩阵相乘的问题，并且用迭代的方法解决了这个问题。他们先假定所有网页的排名是相同的，并且根据这个初始值，算出各个网页的第一次迭代排名，然后再根据第一次迭代排名算出第二次的排名。他们两人从理论上证明了

不论初始值如何选取，这种算法都保证了网页排名的估计值能收敛到排名的真实值。值得一提的事，这种算法是完全没有任何人工干预的。

理论问题解决了，又遇到实际问题。因为互联网上网页的数量是巨大的，上面提到的二维矩阵从理论上讲有网页数量的二次方这么多个元素。如果假定有十亿个网页，那么这个矩阵就有一百亿亿个元素。这么大的矩阵相乘，计算量是非常大的。佩奇和布林两人利用稀疏矩阵计算的技巧，大大简化了计算量，并实现了这个网页排名算法。

互联网网页数量的增长使得 PageRank 的计算量越来越大，必须利用多台服务器才能完成。Google 早期时，PageRank 计算的并行化是半手工、半自动的，这样更新一遍所有网页的 PageRank 的周期很长。2003 年，Google 的工程师发明了 MapReduce 这个并行计算的工具，PageRank 的并行计算完全自动化了，这就大大缩短了计算时间，使网页排名的更新周期比以前短了许多。

我到 Google 后，佩奇和我们几个新员工座谈时，讲起他当年和布林是怎么想到网页排名算法的。他说："当时我们觉得整个互联网就像一张大的图，每个网站就像一个节点，而每个网页的链接就像一个弧。我想，互联网可以用一个图或者矩阵描述，我也许可以用这个发现做篇博士论文。"他和布林就这样发明了 PageRank 算法。PageRank 中的 Page 一词在英文里既有网页、书页等意思，也是佩奇的姓氏我们开玩笑讲，为什么这个算法叫"佩奇"算法不叫"布林"算法？

网页排名的高明之处在于它把整个互联网当作一个整体来对待。这无意识中符合了系统论的观点。相比之下，以前的信息检索大多把每一个网页当作独立的个体对待，大部分人当初只注意了网页内容和查询语句的相关性，忽略了网页之间的关系。虽然在佩奇和布林同时代也有一些人在思考如何利用网页之间的联系来衡量网页的质量，但只是摸到一些皮毛，找到一些拼凑的办法，都没有从根本上解决问题。

PageRank 在当时对搜索结果的影响非常大。在 1997-1998 年前后，所有互联网上能找到的搜索引擎，每十条结果只有两三条是相关的、有用的。而还在斯坦福大学实验室里的 Google 当时能做到每十条结果有七八条是相关的。这是一个质的差别，给人的感觉就如同 iPhone 和老式诺基亚手机的差异那么大。这使得 Google 能够迅速打败以前所有的搜索引擎。但是今天，任何商业的搜索引擎，十条结果都有七八条是相关的了，这时一个新的搜索引擎在技术上投入再大，可提升的空间却非常有限，用户很难感觉到差别。这也是后来微软很难在搜索上有所作为的原因。

2 延伸阅读：PageRank 的计算方法

读者知识背景：线性代数。

假定向量

$$B = (b_1, b_2, \dots, b_N)^{\mathrm{T}} \quad (10.1)$$

为第一、第二、…第 N 个网页的网页排名。矩阵

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} & \cdots & a_{1M} \\ \cdots & & & & \cdots \\ a_{m1} & \cdots & a_{mn} & \cdots & a_{mM} \\ \cdots & & & & \cdots \\ a_{M1} & \cdots & a_{Mn} & \cdots & a_{MM} \end{bmatrix} \quad (10.2)$$

为网页之间链接的数目，其中 a_{mn} 代表第 m 个网页指向第 n 个网页的链接数。A 是已知的，B 是未知的，是我们所要计算的。

假定 B_i 是第 i 次迭代的结果，那么

$$B_i = A \cdot B_{i-1} \quad (10.3)$$

初始假设：所有网页的排名都是 $1/N$，即

$B_0 = \left(\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N}\right)$。

显然通过（10.3）简单（但是计算量非常大）的矩阵运算，可以得到 $B_1, B_2, \ldots$。可以证明（省略）B_i 最终会收敛，即 B_i 无限趋近于 B，此时：$B = B \times A$。因此，当两次迭代的结果 B_i 和 B_{i-1} 之间的差异非常小，接近于零时，停止迭代运算，算法结束。一般来讲，只要 10 次左右的迭代基本上就收敛了。

由于网页之间链接的数量相比互联网的规模非常稀疏，因此计算网页的网页排名也需要对零概率或者小概率事件进行平滑处理。网页的排名是个一维向量，对它的平滑处理只能利用一个小的常数 α。这时，公式（10.3）变成

$$B_i = \left[\frac{\alpha}{N} \cdot I + (1-\alpha) A \right] \cdot B_{i-1} \tag{10.4}$$

其中 N 是互联网网页的数量，α 是一个（较小的）常数，I 是单位矩阵。

网页排名的计算主要是矩阵相乘，这种计算很容易分解成许多小任务，在多台计算机上并行。矩阵相乘具体的并行化方法会在最后介绍 Google 并行计算工具 MapReduce 时再作讨论。

3 小结

今天，Google 搜索引擎比最初复杂、完善了许多。但是 PageRank 在 Google 所有算法中依然是至关重要的。在学术界，这个算法被公认为是文献检索中最大的贡献之一，并且被很多大学列入信息检索课程（Information Retrieval）的教程。佩奇本人也因为这个算法在 30 岁时当选为美国工程院院士，是继乔布斯和盖茨之后又一位当选院士的辍学生。由于 PageRank 算法受到专利保护，它带来两个结果。首先，其他搜索引擎开始时都比较遵守游戏规则，不去侵犯它，这对当时还很弱小的 Google 是一个很好的保护。第二，它使得斯坦福大学拥有了超过 1% 的 Google 股票，带来了超过 10 亿美元的收益。

参考文献：

1. Sergey Brin and Lawrence Page , The Anatomy of a Large-Scale Hypertextual Web Search Engine,http://infolab.stanford.edu/~backrub/google.html

第11章 如何确定网页和查询的相关性

前面已经谈过了如何自动下载网页、如何建立索引、如何衡量网页的质量（PageRank）。接下来谈谈如何确定一个网页和某个查询的相关性。了解了这四个方面，有一定编程基础的读者就可以写出一个简单的搜索引擎了，比如为自己所在的学校或院系搭建一个小型搜索引擎。

我们还是看看前面介绍的例子，查找关于“原子能的应用”的网页。第一步是在索引中找到包含这三个词的网页(详见第8章关于布尔运算的内容)。现在任何一个搜索引擎能提供几十万甚至是上百万个与这个查询词组多少有点关系的网页，比如Google返回了大约一千万个结果。那么哪个应该排在前面呢？显然应该把网页本身质量好，且网页和查询关键词“原子能的应用”相关性高的网页排在前面。第10章已经介绍了如何度量网页的质量。这里介绍另外一个关键技术：如何度量网页和查询的相关性。

1 搜索关键词权重的科学度量 TF-IDF

短语“原子能的应用”可以分成三个关键词：原子能、的、应用。根据直觉，我们知道，包含这三个词较多的网页应该比包含它们较少的网页相关。当然，这个办法有一个明显的漏洞，那就是内容长的网页比内容短的网页占便宜，因为长的网页总的来讲包含的关键词要多些。因此，需要根

据网页的长度，对关键词的次数进行归一化，也就是用关键词的次数除以网页的总字数。我们把这个商称为“关键词的频率”，或者“单文本词频”（Term Frequency），比如，某个网页上一共有1000词，其中“原子能”、“的”和“应用”分别出现了2次、35次和5次，那么它们的词频就分别是0.002、0.035和0.005。将这三个数相加，其和0.042就是相应网页和查询“原子能的应用”的“单文本词频”。

因此，度量网页和查询的相关性，有一个简单的方法，就是直接使用各个关键词在网页中出现的总词频。具体地讲，如果一个查询包含N个关键词$w_1, w_2, \ldots, w_N$，它们在一个特定网页中的词频分别是：$TF_1, TF_2, \ldots, TF_N$。（TF：Term Frequency，是词频一词的英文缩写）。那么，这个查询和该网页的相关性（即相似度）就是：

$$TF_1 + TF_2 + \ldots + TF_N \quad (11.1)$$

读者可能已经发现了又一个漏洞。在上面的例子中，“的”这个词占了总词频的80%上，而它对确定网页的主题几乎没什么用处。我们称这种词叫“停止词”（Stop Word），也就是说，在度量相关性时不应考虑它们的频率。在汉语中，停止词还有“是”、“和”、“中”、“地”、“得”等几十个。忽略这些停止词后，上述网页和查询的相关性就变成了0.007，其中“原子能”贡献了0.002，“应用”贡献了0.005。

细心的读者可能还会发现另一个小漏洞。在汉语中，“应用”是个很通用的词，而“原子能”是个很专业的词，后者在相关性排名中比前者重要。因此，需要对汉语中的每一个词给一个权重，这个权重的设定必须满足下面两个条件：

1. 一个词预测主题的能力越强，权重越大，反之，权重越小。在网页中看到“原子能”这个词，或多或少能了解网页的主题。而看到“应用”一词，则对主题基本上还是一无所知。因此，“原子能”的权重就应该比应用大。

2. 停止词的权重为零。

很容易发现，如果一个关键词只在很少的网页中出现，通过它就容易锁定搜索目标，它的权重也就应该大。反之，如果一个词在大量网页中出现，看到它仍然不很清楚要找什么内容，因此它的权重就应该小。

概括地讲，假定一个关键词 w 在 D_w 个网页中出现过，那么 D_w 越大，w 的权重越小，反之亦然。在信息检索中，使用最多的权重是“逆文本频率指数”（Inverse Document Frequency，缩写为 IDF），它的公式为 $\log\left(\frac{D}{D_w}\right)$，其中 D 是全部网页数。比如，假定中文网页数是 D = 10 亿，停止词“的”在所有的网页中都出现，即 D_w = 10 亿，那么它的 $IDF = \log$ (10 亿 / 10 亿) = $\log(1) = 0$。假如专用词“原子能”在 200 万个网页中出现，即 D_w = 200 万，则它的权重 $IDF = \log(500) = 8.96$。又假定通用词“应用”出现在五亿个网页中，它的权重 $IDF = \log(2)$，则只有 1。

也就是说，在网页中找到一个“原子能”的命中率（Hits）相当于找到九个“应用”的命中率。利用 IDF，上述相关性计算的公式就由词频的简单求和变成了加权求和，即

$$TF_1 \cdot IDF_1 + TF_2 \cdot IDF_2 + \cdots + TF_N \cdot IDF_N \qquad (11.2)$$

在上面的例子中，该网页和“原子能的应用”的相关性为 0.0161，其中“原子能”贡献了 0.0126，而“应用”只贡献了 0.0035。这个比例和我们的直觉比较一致了。

TF-IDF（Term Frequency / Inverse Document Frequency）的概念被公认为信息检索中最重要的发明。在搜索、文献分类和其他相关领域有着广泛的应用。讲起 TF-IDF 的历史蛮有意思。IDF 的概念最早是剑桥大学的斯巴克·琼斯[1]（Karen Spärck Jones）提出来的。斯巴克·琼斯 1972 年在一篇题为“关键词特殊性的统计解释和它在文献检索中的应用”的论文中提出 IDF 的概念。遗憾的是，她既没有从理论上解释为什么权

1 斯巴克·琼斯，剑桥大学计算机女科学家，最著名的言论：“计算机是如此重要，因此不能只留给男人去做！”在程序界广为流传。

重 IDF 应该是对数函数 $\log\left(\frac{D}{D_w}\right)$（而不是其他函数，比如平方根 $\sqrt{\frac{D}{D_w}}$），也没有在这个题目上作进一步的深入研究，以至于在以后的很多文献中人们提到 TF-IDF 时没有引用她的论文，绝大多数人甚至不知道斯巴克·琼斯的贡献。同年剑桥大学的罗宾逊写了一个两页纸的解释，解释得很不好。倒是后来康奈尔大学的萨尔顿（Salton）多次撰文、写书讨论 TF-IDF 在信息检索中的用途，加上萨尔顿本人的大名（信息检索领域的世界级大奖就是以萨尔顿的名字命名的），很多人都引用萨尔顿的书，甚至以为这个信息检索中最重要的概念是他提出的。当然，世界并没有忘记斯巴克·琼斯的贡献。2004 年，在纪念《文献学学报》创刊 60 周年之际，该学报重印了斯巴克·琼斯的大作。罗宾逊在同期期刊上写了篇文章，用香农的信息论解释 IDF，这回的解释是对的，但文章写得并不好，非常冗长（足足 18 页），把简单问题搞复杂了。其实，信息论的学者们已经发现并指出，所谓 IDF 的概念就是一个特定条件下关键词的概率分布的交叉熵（Kullback-Leibler Divergence）（详见本书第 6 章“信息的度量和作用”）。这样，关于信息检索相关性的度量，又回到了信息论。

现在的搜索引擎对 TF-IDF 进行了不少细微的优化，使得相关性的度量更加准确了。当然，对有兴趣写一个搜索引擎的爱好者来讲，使用 TF-IDF 就足够了。如果结合网页排名（PageRank）算法，那么给定一个查询，有关网页的综合排名大致由相关性和网页排名的乘积决定。

2 延伸阅读：TF-IDF 的信息论依据

读者背景知识：信息论和概率论。

一个查询（Query）中每一个关键词（Key Word）w 的权重应该反映这个词对查询来讲提供了多少信息。一个简单的办法就是用每个词的信息量作为它的权重，即

$$I(w) = -P(w)\log P(w)$$

$$= -\frac{TF(w)}{N}\log\frac{TF(w)}{N} = \frac{TF(w)}{N}\log\frac{N}{TF(w)} \quad (11.3)$$

其中，N 是整个语料库的大小，是个可以省略的常数。上面的公式可以简化成

$$I(w) = TF(w)\log\frac{N}{TF(w)} \quad (11.4)$$

但是，公式（11.4）有一个缺陷：两个词出现的频率 TF 相同，一个是某篇特定文章中的常见词，而另外一个词是分散在多篇文章中，那么显然第一个词有更高的分辨率，它的权重应该更大。显然，更好的权重公式应该反映出关键词的分辨率。

如果做一些理想的假设，

1）每个文献大小基本相同，均为 M 个词，即 $M = \frac{N}{D} = \frac{\sum_w TF(w)}{D}$。

2）一个关键词在文献一旦出现，不论次数多少，贡献都等同，这样一个词要么在一个文献中出现 $c(w) = \frac{TF(w)}{D(w)}$ 次，要么是零。注意，$c(w) < M$，

那么从公式（11.4）出发可以得到下面的公式：

$$TF(w)\log\frac{N}{TF(w)} = TF(w)\log\frac{MD}{c(w)D(w)}$$

$$= TF(w)\log\left(\frac{D}{D(w)}\frac{M}{c(w)}\right) \quad (11.5)$$

这样，我们看到 TF-IDF 和信息量之间的差异就是公式（11.6）中的第二项。因为 $c(w) < M$，所以第二项大于零，它是 $c(w)$ 的递减函数。把上面的公式重写成

$$TF\text{-}IDF(w) = I(w) - TF(w)\log\frac{M}{c(w)} \tag{11.6}$$

可以看到，一个词的信息量 $I(w)$ 越多，TF-IDF 值越大；同时 w 命中的文献中 w 平均出现的次数越多，第二项越小，TF-IDF 也越大。这些结论和信息论完全相符。

3 小结

TF-IDF 是对搜索关键词的重要性的度量，从理论上讲，它有很强的理论根据。因此，如果对搜索不是很精通的人，直接采用 TF-IDF 效果也不会太差。现在各家搜索引擎对关键词重要性的度量，都在 TF-IDF 的基础上有些改进和微调。但是，在原理上与 TF-IDF 相差不远。

参考文献：

1. Spärck Jones, Karen "A statistical interpretation of term specificity and its application in retrieval". Journal of Documentation28 (1): 11-21, 1972

2. Salton, G. and M. J. McGill, *Introduction to modern information retrieval*. McGraw-Hill, 1986

3. H.C. Wu, R.W.P. Luk, K.F. Wong, K.L. Kwok "Interpreting tf idf term weights as making relevance decisions". *ACM Transactions on Information Systems* 26 (3): 1-37, 2008

第12章　地图和本地搜索的最基本技术——有限状态机和动态规划

2007年，第一次在谷歌黑板报上写“数学之美”系列时，本地搜索和服务还不是很普及，智能手机不仅数量有限，而且完全没有结合本地信息。地图服务的流量与网页搜索相比，只是众多垂直搜索的一部分。今天，本地生活服务变得越来越重要，而确认地点、查看地图、查找路线等等依然是本地生活服务的基础。这里为了减少章节的数量，我将有限状态机这个系列和动态规划的一部分合并，组成围绕地图和本地生活搜索的一章。

2008年9月23日，Google、T-Mobile和HTC宣布了第一款基于开源操作系统Android的3G手机G1。这款手机的外观和体验远不如一年之前苹果推出的第一款iPhone，价钱也差不了太多，但是依然有不少人使用。它的杀手级功能是利用全球卫星定位系统实现全球导航。卫星导航的功能早在2000年前后就已有车载设备使用，但是售价昂贵。2004年我购买的一款麦哲伦便携式导航系统，价格在1000美元左右（现在只要两三百美元），后来一些智能手机也开发了这个功能，但是基本上不可用。Android手机的这个功能当时已经完全可以媲美任何一个卫星导航仪，加上它的地址识别技术（采用有限状态机）比任何一个卫星导航仪严格的地址匹配技术（不能输错一个字母）要好得多，结果麦哲伦等导航仪制造公司在G1发布当天的股价暴跌四成。

智能手机的定位和导航功能，里面的关键技术只有三个：第一是利用卫星定位，这一点传统的导航仪都能做到，不做介绍；第二是地址的识别，在本章第一节中介绍；第三，根据用户输入的起点和终点，在地图上规划最短路线或者最快路线，在本章第二节中介绍。

1 地址分析和有限状态机

地址的识别和分析是本地搜索必不可少的技术。判断一个地址的正确性同时非常准确地提炼出相应的地理信息（省、市、街道、门牌号等等）看似简单，实际上很麻烦。比如腾讯公司在深圳的地址，我收到的邮件和包裹上面有如下各种各样的地址：

广东省深圳市腾讯大厦

广东省 518057 深圳市南山区科技园腾讯大厦

深圳市 518057 科技园腾讯大厦

深圳市南山区科技园腾讯公司

深圳市南山区科技园腾讯总部 518000（估计不知道准确的邮编）

广东省深圳市科技园中一路腾讯公司

……

这些地址写得都有点不清楚，但是邮件和包裹我都收到了，说明邮递员可以识别。但是，如果让一个程序员写一个分析器分析这些地址的描述，恐怕就不是一件容易的事了。其根本原因在于，地址的描述虽然看上去简单，但是它依然是比较复杂的上下文有关的文法，而不是上下文无关。比如下面的两个地址：

上海市北京东路 xx 号

南京市北京东路 xx 号

当识别器扫描到“北京东路”时，它和后面的门牌号是否构成一个正确

的地址需要看它的上下文，即城市名。我们在前面的章节中讲过，上下文有关文法的分析既复杂又耗时，它的分析器也不好写。如果没有好的模型，这个分析器写出来很难看不说，还有很多情况覆盖不了。比如这样的地址描述：

（深圳市）深南大道和南山大道交口西 100 米[1]

1 很多商家还真是这么描述它们的地址。

所幸的是，地址的文法是上下文有关文法中相对简单的一种，因此有许多识别和分析的方法，但最有效的是有限状态机。

一个有限状态机是一个特殊的有向图（参见本书第 9 章中图论相关的内容），它包括一些状态（节点）和连接这些状态的有向弧。图 12.1 是一个识别中国地址的有限状态机的简单例子。

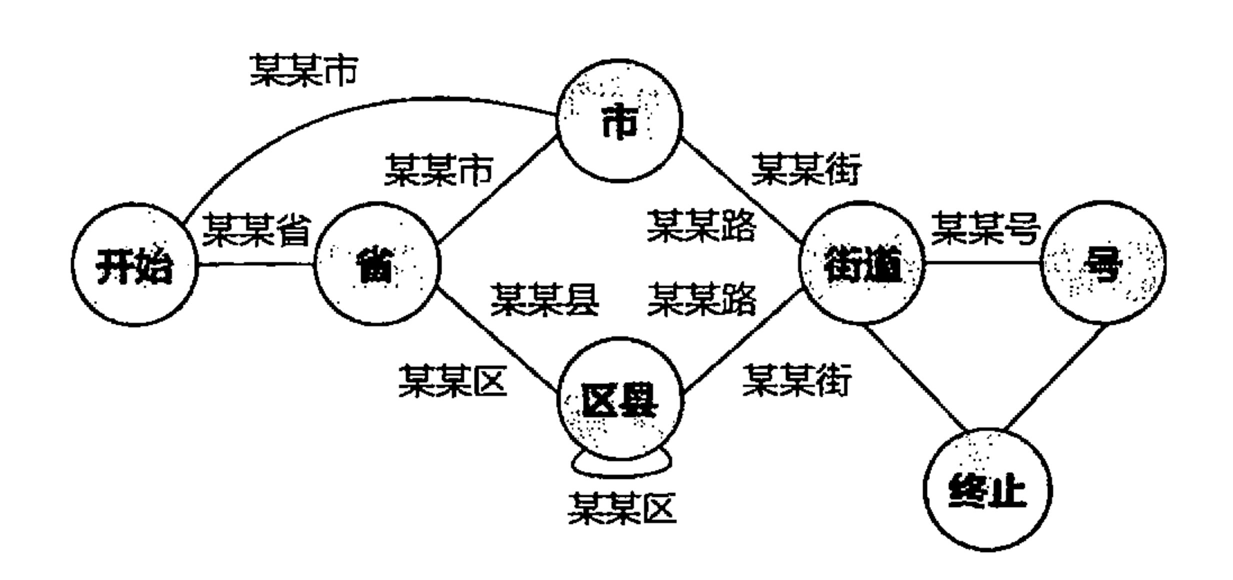


图 12.1 识别地址的有限状态机

每一个有限状态机都有一个开始状态和一个终止状态，以及若干中间状态。每一条弧上带有从一个状态进入下一个状态的条件。比如，在上图中，当前的状态是“省”，如果遇到一个词组和（区）县名有关，就进入状态“区县”；如果遇到的下一个词组和城市有关，那么就进入“市”的状态，如此等等。如果一条地址能从状态机的开始状态经过状态机的若干中间状态，走到终止状态，那么这条地址就有效，否则无效。比如，“北京市双清路 83 号”对于上面的有限状态来讲有效，而“上海市辽宁省马家庄”则无效（因为无法从“市”走回到“省”）。

使用有限状态机识别地址，关键要解决两个问题，即通过一些有效的地址建立状态机，以及给定一个有限状态机后，地址字串的匹配算法。好在这两个问题都有现成的算法。有了关于地址的有限状态机后，就可以用它分析网页，找出网页中的地址部分，建立本地搜索的数据库。同样，也可以对用户输入的查询进行分析，挑出其中描述地址的部分，当然，剩下的关键词就是用户要找的内容。比如，对于用户输入的“北京市双清路附近的酒家”，Google 本地会自动识别出地址“北京市双清路”和要找的对象“酒家”。

上述基于有限状态机的地址识别方法在实用中会有一些问题：当用户输入的地址不太标准或者有错别字时，有限状态机会束手无策，因为它只能进行严格匹配。（其实，有限状态机在计算机科学中早期的成功应用是在程序语言编译器的设计中。一个能运行的程序在语法上必须是没有错的，所以不需要模糊匹配。而自然语言则很随意，无法用简单的语法描述。）

为了解决这个问题，我们希望看到可以进行模糊匹配，并给出一个字串为正确地址的可能性。为了实现这一目的，科学家们提出了基于概率的有限状态机。这种基于概率的有限状态机和离散的马尔可夫链（详见前面关于马尔可夫模型的章节）基本上等效。

在上个世纪 80 年代以前，尽管有不少人使用基于概率的有限状态机，但都是为自己的应用设计专用的有限状态机的程序。上个世纪 90 年代以后，随着有限状态机在自然语言处理上的广泛应用，不少科学家致力于编写通用的有限状态机程序库。其中，最成功的是前 AT&T 实验室的三位科学家，莫瑞（Mehryar Mohri）、皮耶尔（Fernando Pereira）和瑞利（Michael Riley）。他们三人花了很多年时间，编写成一个通用的基于概率的有限状态机 C 语言工具库。由于 AT&T 有对学术界免费提供各种编程工具的好传统，他们三人也把自己多年的心血拿出来和同行们共享。可惜好景不长，AT&T 实验室风光不再，这三个人都离开了 AT&T，莫瑞成了纽

约大学的教授；皮耶尔先当了宾夕法尼亚大学的计算机系主任，而后成为 Google 的研究总监；而瑞利直接成为 Google 的研究员。有一段时间，AT&T 实验室的新东家不再免费提供有限状态机 C 语言工具库。虽然此前莫瑞等人公布了他们的详细算法[2]，但是省略了实现的细节。因此，在学术界，不少科学家能够重写同样功能的工具库，但是很难达到 AT&T 工具库的效率（即运算速度），这一度是一件令人遗憾的事。但是近年来，随着开源软件在世界上的影响力越来越大，AT&T 又重新开放了这个工具的源代码。有限状态机的程序不是很好写，它要求编程者既懂得里面的原理和技术细节，又要有很强的编程能力，因此建议大家直接采用开源的代码就好。

2 http://www.cs.nyu.edu/~mohri/pub/csl01.pdf

2　全球导航和动态规划

全球导航的关键算法是计算机科学图论中的动态规划（Dynamic Programming）的算法。

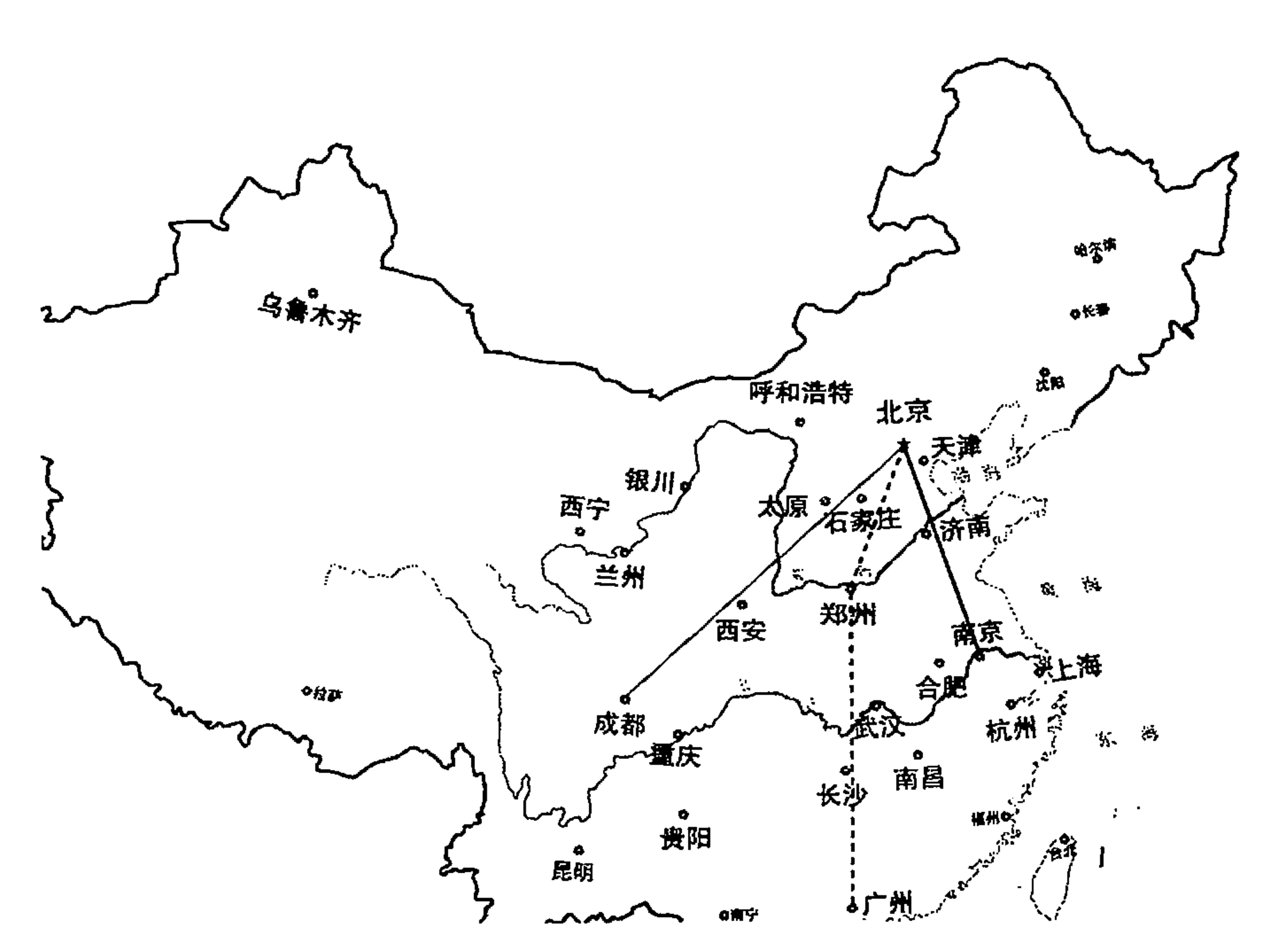


图 12.2　中国公路图是一个特殊的加权图

在图论中，一个抽象的图包括一些节点和连接它们的弧。如果再考虑每条弧的长度，或者说权重，那么这个图就是加权图（Weighted Graph）。比如说中国公路网就是一个很好的“加权图”例子：每个城市是一个节点，每一条公路是一条弧。图中弧的权重对应于地图上的距离，或者是行车时间、过路费金额，等等。图论中很常见的一个问题是要找一个图中给定两个点之间的最短路径（Shortest Path）。比如，想找到从北京到广州的最短行车路线或者最快行车路线。当然，最直接的笨办法是把所有可能的路线看一遍，然后找到最优的。这种办法只有在节点数是个位数的图中还行得通，当图的节点数（城市数目）有几十个的时候，计算的复杂度就已经让人甚至计算机难以接受了，因为所有可能路径的个数随着节点数的增长而呈指数（或者说几何级数）增长，也就是说，每增加一个城市，复杂度要大一倍。显然导航系统不会用这种笨办法——任何导航仪或者导航软件都能在几秒钟内就找到最佳行车路线。

所有的导航系统采用的都是动态规划的办法，这里面的 Programming 一词在数学上的含义是“规划”，不是计算机里的“编程”。动态规划的原理其实很简单。以上面的问题为例，当我们要找从北京到广州的最短路线时，先不妨倒过来想这个问题：假如已经找到了所要的最短路线（称为路线一），如果它经过郑州，那么从北京到郑州的这条子路线（比如是北京→保定→石家庄→郑州，暂定为子路线一），必然也是所有从北京到郑州的路线中最短的。否则，可以假定还存在从北京到郑州更短的路线（比如北京→济南→徐州→郑州，暂定为子路线二），那么只要用这第二条子路线代替第一条，就可以找到一条从北京到广州全程更短的路线（称为路线二），这就和我们讲的路线一是北京到广州最短的路线相矛盾。其矛盾的根源在于，假设的子路线二或者不存在，或者比子路线一还来得长。

在实际实现这个算法时，我们又正过来解决这个问题，也就是说，要想找到从北京到广州的最短路线，先要找到从北京到郑州的最短路线。当然，聪明的读者可能已经发现其中的一个“漏洞”，就是在还没有找到全程

最短路线前，不能肯定它一定经过郑州。不过没有关系，只要在图上横切一刀，这一刀要保证将任何从北京到广州的路线一分为二，如图 12.3。

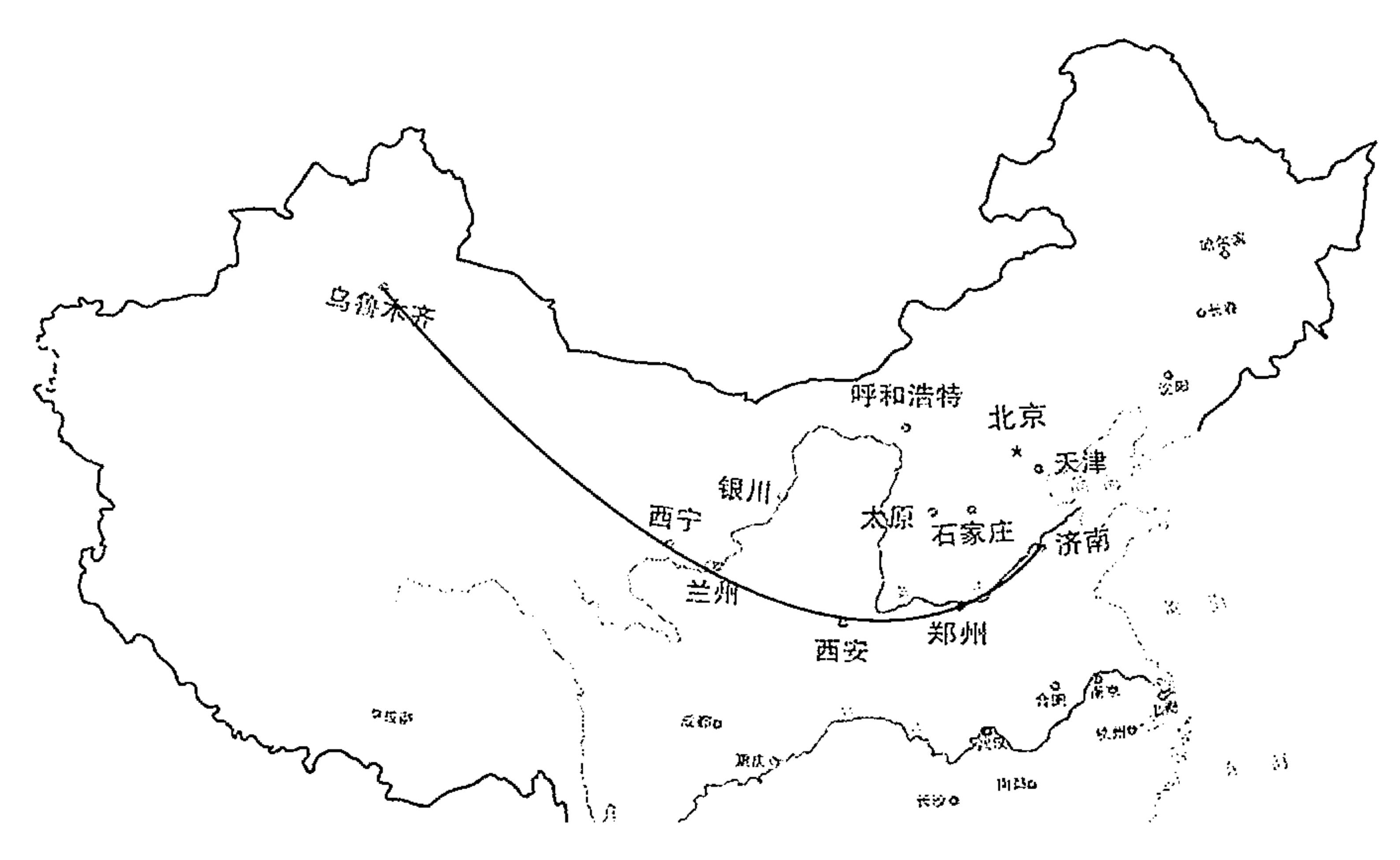


图 12.3　从北京到广州的路线必须经过图中粗线上的某个城市

那么从广州到北京的最短路径必须经过这一条线上的某个城市（乌鲁木齐、西宁、兰州、西安、郑州、济南）。我们可以先找到从北京出发到这条线上所有城市的最短路径，最后得到的全程最短路线一定包括这些局部最短路线中的一条，这样，就可以将一个“寻找全程最短路线”的问题，分解成一个个寻找局部最短路线的小问题。只要将这条横切线从北京向广州推移，直到广州为止，我们的全程最短路线就找到了。这便是动态规划的原理。采用动态规划可以大大降低最短路径的计算复杂度。在上面的例子中，每加入一条横切线，线上平均有 10 个城市，从广州到北京最多经过 15 个城市，那么采用动态规划的计算量是 10 × 10 × 15，而采用穷举路径的笨办法是 10 的 15 次方，前后差了万亿倍。

正确的数学模型可以将一个计算量看似很大的问题的计算复杂度大大降低。这便是数学的妙用。

3 延伸阅读：有限状态传感器

读者背景知识：图论。

有限状态机的应用远不止地址的识别，今天的语音识别解码器基本上是基于有限状态机的原理。另外，它在编译原理、数字电路设计上有着非常重要的应用，因此在这里给出有限状态机严格的数学模型。

定义（有限状态机）：有限状态机是一个五元组$(\Sigma, S, s_0, \delta, f)$，其中：

Σ是输入符号的集合，

S是一个非空的有限状态集合，

s_0是S中的一个特殊状态，起始状态，

δ是一个从空间$S \times \Sigma$到S的映射函数，即$\delta: S \times \Sigma \to S$。

f是S中另外一个特殊状态，终止状态。

这里面映射函数δ对于一些变量，即状态和输入符号的组合可能没有合适的对应状态（函数值），也就是说，在一些状态下，有的符号不能被接收，比如在12.1的例子中，进入到城市这个状态后，对于“省份”这样的输入就进入不了新的状态了。这时，有限状态机就会发出出错信号。如果一个状态序列在有限状态机中能够从s_0开始经过一些状态进入终止状态f，那么它就是可以由这个有限状态机生成的合法序列，反之则不是。

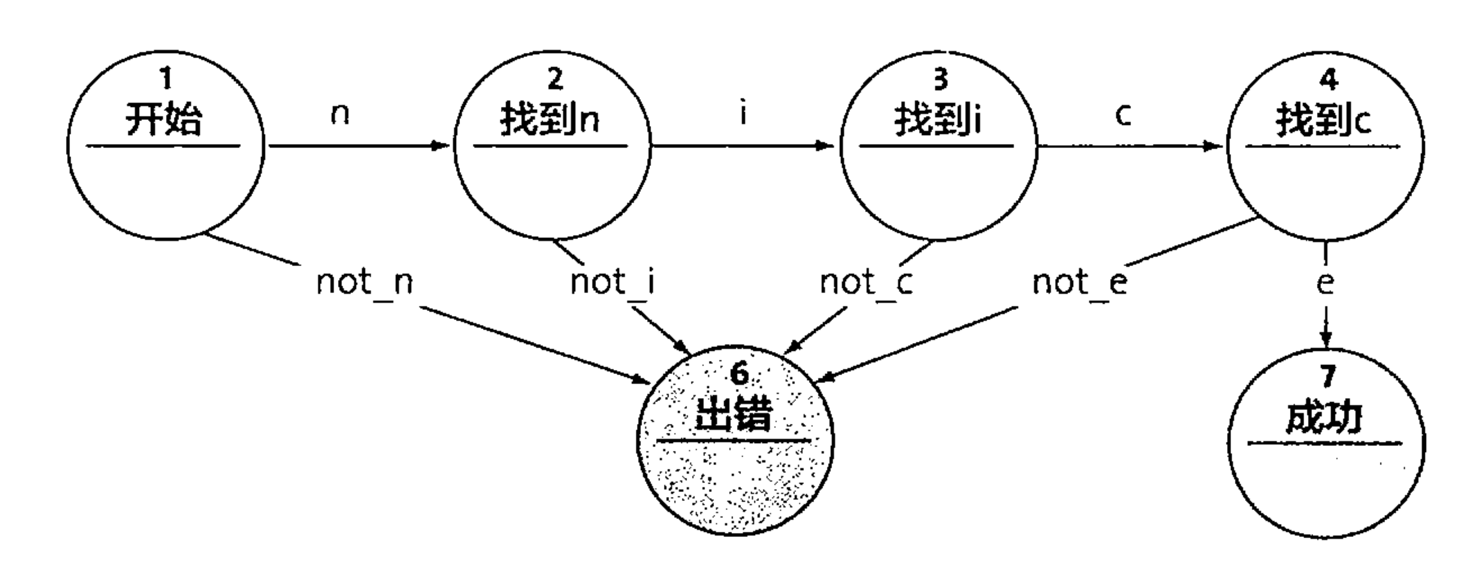


图12.4 有限状态机中一些状态和输入符号的组合没有合适的对应状态，这时可以把它们对应到出错状态

有限状态机在语音识别和自然语言理解中起着非常重要的作用，不过这些领域使用的是一种特殊的有限状态机 —— 加权的有限状态传感器（Weighted Finite State Transducer，简称 WFST）。下面介绍 WFST 及其构造和用法。

有限状态传感器（Finite State Transducer）的特殊性在于，有限状态机中的每个状态由输入和输出符号定义。如图 12.5。

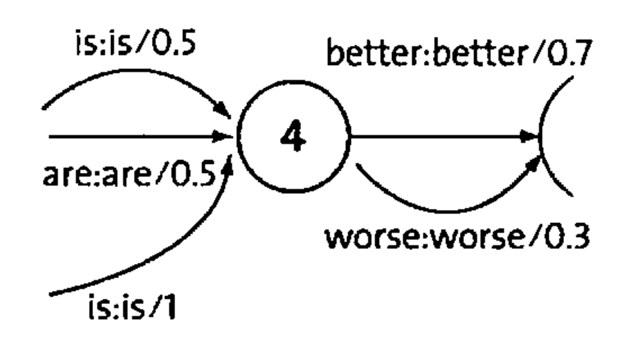


图 12.5　有限状态传感器

状态 4 的定义是"输入为 is 或者 are，输出为 better 或者 worse"的状态。不管整个符号序列前后如何，只要在某一时刻前后的符号为 is / are 和 better / worse 的组合，就能进入此状态。状态可以有不同输入和输出，如果这些输入和输出的可能性不同，即赋予了不同的权重，那么相应的有限状态传感器就是加权的。对比我们在第 2 章"自然语言处理"中提到的二元模型，读者可能会发现任何一个词的前后二元组，都可以对应到 WFST 的一个状态。因此，WFST 是天然的自然语言处理的分析工具和解码工具。

在语音识别中，每个被识别的句子都可以用一个 WFST 来表示。

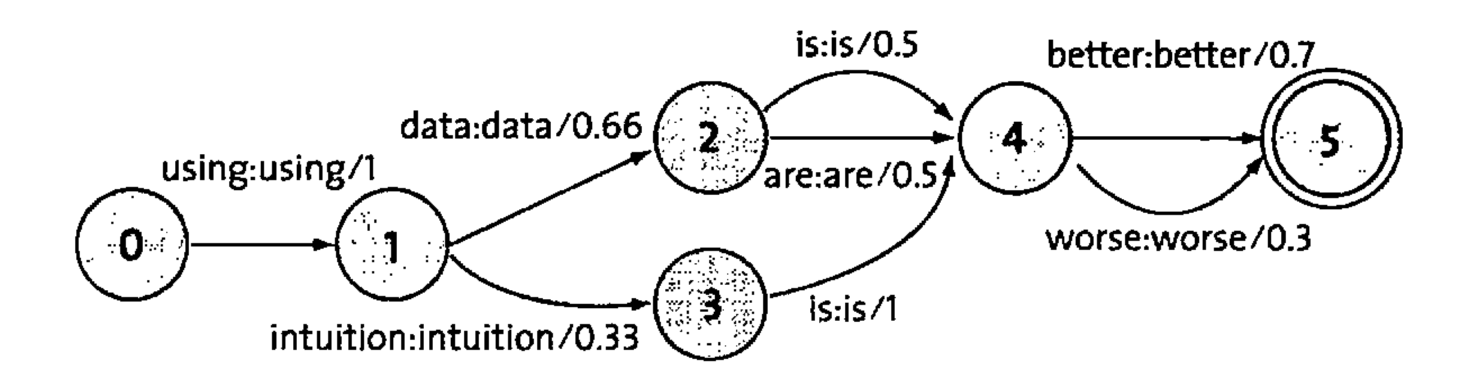


图 12.6　描述的一句话的语音的识别结果，句子的意思是"使用数据好，使用直觉不好"

WFST 中的每一条路径就是一个候选的句子，其中概率最大的那条路径就是这个句子的识别结果。而这个算法的原理是上节介绍的动态规划。

4 小结

有限状态机和动态规划的应用非常广泛，远远不止识别地址、导航等地图服务相关领域。它们在语音识别、拼写和语法纠错、拼音输入法、工业控制和生物的序列分析等领域都有着极其重要的应用。其中在拼音输入法中的应用后面还要再介绍。

参考文献：

1. MehryarMohri, Fernando Pereira, Michael Riley, Weighted finite-state transducers in speech recognition, Computer Speech and Language, V16-1, pp69-88,2002

第13章 Google AK-47 的设计者
—— 阿米特·辛格博士

枪迷或者看过尼古拉斯·凯奇(Nicolas Cage)主演的电影“战争之王”(Lord of War)的人也许还记得影片开头的一段话：(在所有轻武器中)最有名的是AK-47冲锋枪(也就是中国五六式冲锋枪的原型，全世界共制造了7 500万支，另外制造了一亿支“兼容”的)，因为它从不卡壳，不易损坏，可在任何环境下使用，可靠性好，杀伤力大并且操作简单。



图13.1 AK-47冲锋枪

我认为，在计算机科学领域，一个好的算法应该像AK-47冲锋枪那样：简单、有效、可靠性好而且容易读懂(或者说易操作)，而不应该是故弄玄虚。Google Fellow、美国工程院院士阿米特·辛格博士(Amit Singhal)就是Google AK-47的设计者，在公司内部，Google的排序算法Ascorer里面的A便是他的名字首字母。

从加入 Google 的第一天，我就开始了和辛格四年愉快的合作，而他一直是我的良师益友。辛格、马特・柯茨（Matt Cutts，中国一些用户误认为他是联邦调查局特工，当然他不是）、马丁・柯斯尔基（Martin Kaszkiel）和我四个人一同研究和解决网络搜索中的作弊问题（Spam）[1]。我们发现绝大多数作弊的搜索都多少有些商业意图。这也合情合理，因为利益使然。因此，我们需要建一个分类器，准确区分一个搜索是否有商业意图。我以前一直在学术界学习和工作，凡事倾向于寻找完美的解决方案。设计一个可用的、漂亮的分类器对我来讲不是难事，但是实现和训练它大约要花三四个月的时间，当时 Google 还没有 MapReduce 这种并行计算工具，复杂的机器学习非常耗时。而辛格认为找个简单有效的办法就行了，他问我实现一个最简单可用的分类器大约需要多少时间，我说一个周末[2]可能就够了。周一我把分类器完成了，问他是否还需要花时间去实现一个完美的方案。辛格看了看结果说，“够好了，够好了，在工程上简单实用的方法最好。”我们就依照这个原则，对其他问题也是找简单实用的方案，结果一两个月就把作弊的数量减少了一半。当时我们和公司工程副总裁韦恩・罗森（Wayne Rosing）打了个赌，如果我们能减少 40% 的作弊，他就给我们发工程奖，送我们四个家庭（不止是四个员工）去夏威夷度假五天。这个反作弊的算法上线后，罗森真的履约了。谢尔盖・布林问我怎么在这么短的时间里实现这么些功能。我告诉他其实方法很简单。布林讲，“哦，那就像 AK-47 自动步枪。”这个分类器设计得非常小巧（只用很小的内存），而且非常快速（几台服务器就能处理全球搜索的分类），至今运行得很好，即使在我离开 Google 后，依然在使用。这项技术，也是 Google 取得的第一个反作弊方面的美国专利。

1 对于搜索反作弊，以后还要专门讲述。

2 在 Google 创业时期，我们周末一般是不休息的。

后来我和辛格一起又完成了许多项目，包括对中、日、韩文排名新算法的设计和实现。在 2002 年，Google 虽然支持对 70 种语言的检索，但是所有的语言只有一个排名算法。当时的国际化工作仅仅局限于翻译界面和字符编码的适应。辛格找我来一起做一个全新的中、日、韩文搜索算法。说实话，我当时对特定语言的搜索不感兴趣，但是公司只有我一个

学自然语言处理的中国人，而当时的中日韩搜索结果相比英文又很“烂”，这件事便落到了我的头上。有了上次的经验，我这次也干脆直接用了一个“简单”的方案。这个方法效果虽然很好，但是占用内存较多，当然Google 的服务器数量还没有现在这么多，不可能为了中日韩这三个占总流量不到 10% 的语言额外增加一批服务器。辛格提出用一个拟合函数替代很耗内存的语言模型，这样不需要增加任何服务器。但是，这样一来搜索质量的提高幅度只有原来采用大模型时的 80%。我对此多少有点不甘心。辛格解释说，这样可以让我们至少早两个月将这个新算法提供给中国的用户，他们的用户体验将比现在有质的提高，这是雪中送炭。我们暂时放弃掉的 20% 收益，对他们来讲是锦上添花的事。我接受了他的建议，在 2003 年初我发布了第一个专门为中日韩语言设计的搜索算法。一年后，Google 的服务器数量也有所增加。我在模型压缩上也有了进步，这时便发布了完整的中日韩语言搜索算法。辛格这种做事情的哲学，即先帮助用户解决 80% 的问题，再慢慢解决剩下的 20% 问题，是在工业界成功的秘诀之一。许多失败并不是因为人不优秀，而是做事情的方法不对，一开始追求大而全的解决方案，之后长时间不能完成，最后不了了之。

在 Google 里，辛格一直坚持寻找简单有效的解决方案，因为他奉行简单的哲学。但是这种做法在 Google 这个人才济济的公司里常常招人反对，因为很多资深的工程师倾向于低估简单方法的有效性。2003-2004 年，Google 从世界上很多知名实验室和大学，比如 MITRE[3]、AT&T 实验室和 IBM 实验室，招揽了不少自然语言处理的科学家。不少人试图用精确而复杂的办法对辛格设计的各种“AK-47”进行改进，后来发现几乎所有时候，辛格的简单方法都接近最优的解决方案，而且还快得多。这些“徒劳者”中包括两个世界级的人物，乌迪 · 曼波（Udi Manber）和费尔南多 · 皮耶尔（Fernando Pereira）。

3 世界著名的情报处理实验室，主要为美国国防部、国家安全局（NSA）等机构进行情报处理的研究以及保密情报（Classified）的处理。

2006 年夏天，乌迪 · 曼波加盟 Google。曼波是世界上最早一批研究搜索的学者，之前是大学教授、雅虎的首席科学家和首席算法官（Chief Algorithm Officer，这个职务有点无聊）和亚马逊搜索引擎 A9 的 CEO。

曼波一来就召集了十几名科学家和工程师，利用机器学习的方法来改进辛格的那些简单模型。经过半年的努力，曼波发现他似乎是在做无用功，一年后彻底放弃了这方面的努力，从此转向人事管理工作。2008 年，Google 花了很大代价动员了世界著名的语音识别和自然语言处理专家、宾夕法尼亚大学计算机系主任费尔南多·皮耶尔加盟。皮耶尔是辛格在 AT&T 时的直接上司，著名的有限状态机 AT&T FST 工具的作者之一。皮耶尔和辛格对做好计算机搜索的认识完全不同。皮耶尔认为最好的计算机搜索算法一定要先理解文本的意思，然后才能准确检索。因此，提高搜索质量的关键是文本的句法分析。辛格认为，计算机不必学习人的做法，就如同飞机不必像鸟一样飞行。在 Google 内部两人的关系已经反了个个儿，辛格成了唱主角的。他原本希望皮耶尔能在他自己的基础上帮他更上一层楼，但是皮耶尔的技术路线和辛格明显不同。碍于老上司的情面，辛格只好由着皮耶尔按照自己的想法做。布林考虑到花了很大代价请来皮耶尔，在资源上也给了他足够的支持。但是，三年下来皮耶尔的团队还是毫无建树。这可能也可以解释为什么微软投入了几十亿美元在搜索上依然难以超过 Google。

辛格坚持选择简单方案的另一个原因是容易解释每一个步骤和方法背后的道理，这样不仅便于出了问题时查错（Debug），而且容易找到今后改进的目标。今天，整个业界的搜索质量比十多年前佩奇和布林开始研究搜索时已经有了很大的提高，大的改进之处已经不存在了。现在几乎所有的改进都非常细微：通常对一类搜索有改进的方法，会对另外某一类搜索产生稍稍负面的影响。这时候，需要很清楚“所以然”，才能找出这个方法产生负面影响的原因和场景，并且避免它的发生。对于非常复杂的方法，尤其是像黑盒子似的基于机器学习的方法，这一点是做不到的。而如果每一项改进都是有得有失，甚至得失相差无几，那么长期下来搜索的质量不会有什么明显提升。辛格要求对于搜索质量的改进方法都要能说清楚理由，说不清楚理由的改进即使看上去有效也不会采用，因为这样将来可能是个隐患。这一点和微软、雅虎把搜索质量提升当作一个黑

盒子完全不同。辛格的做法基本上能保证 Google 搜索的质量长期来讲是稳步提高。

当然，辛格之所以总是能找到那些简单有效的方法，不是靠直觉，更不是撞大运，这首先是靠他丰富的研究经验。辛格早年师从于搜索大师萨尔顿（Salton）教授，毕业后就职于 AT&T 实验室。在那里，他和两个同事半年就搭建了一个中等规模的搜索引擎，这个引擎索引的网页数量虽然无法和商用引擎相比，但是准确性却非常好。早在 AT&T，他就对搜索问题的各个细节进行了仔细的研究，他的那些简单而有效的解决方案，常常是深思熟虑去伪存真的结果。其次，辛格坚持每天要分析一些搜索结果不好的例子，以掌握第一手的资料，即使在他成为 Google Fellow 以后，依旧如此。这一点，非常值得从事搜索研究的年轻工程师学习。事实上，我发现中国大部分做搜索的工程师在分析不好的结果上花的时间远比功成名就的辛格要少。

辛格非常鼓励年轻人要不怕失败，大胆尝试。有一次，一位刚毕业不久的工程师因为把带有错误的程序推出到 Google 的服务器上而惶惶不可终日。辛格安慰她说，你知道，我在 Google 犯的最大一次错误是曾经将所有网页的相关性得分全部变成了零，于是所有搜索的结果全部是随机的了。这位工程师后来为 Google 开发了很多好产品。

辛格在 AT&T 实验室时确立了他在学术界的地位，但是，他不满足于只是做实验写论文，于是他离开实验室，来到当时只有百十号人的 Google。在这里，他得以施展才智，重写了 Google 的排名算法，并且一直在负责改进它。辛格因为舍不得放下两个孩子，很少参加各种会议，但是他仍然被学术界公认为是当今最权威的网络搜索专家。2005 年，辛格作为杰出校友被请回母校康奈尔大学计算机系，在 40 周年系庆上作报告。获得这一殊荣的还有大名鼎鼎的美国工程院院士，计算机独立磁盘冗余阵列（RAID）的发明人凯茨（Randy Katz）教授。

辛格和我在性格和生活习惯上有很大的不同，他基本上是个素食者，至

少过去我不是很喜欢他家的饭菜。但是我们有一点非常相似，就是遵循简单的哲学。

2012 年，辛格当选美国工程院院士，并出任主管 Google 搜索的高级副总裁。

第14章 余弦定理和新闻的分类

世界上有些事情常常超乎人们的想象。余弦定理和新闻的分类似乎是两件八杆子打不着的事，但是它们确有紧密的联系。具体地说，新闻的分类很大程度上依靠余弦定理。

2002年夏天，Google推出了自己的“新闻”服务。和传统媒体的做法不同，这些新闻不是记者写的，也不是人工编辑的，而是由计算机整理、分类和聚合各个新闻网站的内容，一切都是自动生成的。这里面的关键技术就是新闻的自动分类。

1 新闻的特征向量

所谓新闻的分类，或者更广义地讲任何文本的分类，无非是要把相似的新闻放到同一类中。如果让编辑来对新闻分类，他一定是先把新闻读懂，然后找到它的主题，最后根据主题的不同对新闻进行分类。但是计算机根本读不懂新闻，虽然一些商业人士和爱炫耀自己才学的计算机专家宣称计算机能读懂新闻。计算机本质上只能做快速计算。为了让计算机能够“算”新闻（而不是读新闻），就要求我们首先要把文字的新闻变成可以计算的一组数字，然后再设计一个算法来算出任意两篇新闻的相似性。

首先让我们来看看怎样找一组数字（或者说一个向量）来描述一篇新闻。新闻是传递信息的，而词是信息的载体，新闻的信息和词的语义是联系在一起的。套用俄罗斯文豪托尔斯泰在《安娜·卡列尼娜》开篇的那句话[1]来讲，“同一类新闻用词都是相似的，不同类的新闻用词各不相同”。当然，一篇新闻有很多词，有些词表达的语义重要，有些相对次要。那么如何确定哪些重要，哪些次要呢？首先，直觉告诉我们含义丰富的实词一定比“的、地、得”这些助词，或者“之乎者也”这样的虚词重要，这点是肯定的。接下来，需要进一步对每个实词的重要性进行度量。回忆一下我们在“如何确定网页和查询的相关性”一章中介绍的单文本词汇频率／逆文本频率值 TF-IDF 的概念，在一篇文章中，重要的词 TF-IDF 值就高。不难想象，和新闻主题有关的那些实词频率高，TF-IDF 值很大。

1
“幸福的家庭都是相似的，不幸的家庭各有各的不幸”。

现在我们找到了一组来描述新闻主题的数字：对于一篇新闻中的所有实词，计算出它们的 TF-IDF 值。把这些值按照对应的实词在词汇表的位置依次排列，就得到一个向量。比如，词汇表中有 64 000 个词[2]，其编号和词如表 14.1 所示。

2
如果把词典的大小限制在 65 535 词以内，在计算机中只要用两个字节就能表示一个词。

表 14.1 统计词汇表

单词 编号	汉字词
1	阿
2	啊
3	阿斗
4	阿姨
…	…
789	服装
…	…
64000	做作

在某一篇特定的新闻中，这 64 000 个词的 TF-IDF 值分别如表 14.2 所示：

表 14.2 某篇新闻对应的 TF-IDF 值

单词编号	TF-IDF 值
1	0
2	0.0034
3	0
4	0.00052
…	…
789	0.034
…	…
64000	0.075

如果单词表中的某个词在新闻中没有出现，对应的值为零，那么这 64 000 个数，组成一个 64 000 维的向量。我们就用这个向量来代表这篇新闻，并成为新闻的特征向量（Feature Vector）。每一篇新闻都可以对应这样一个特征向量，向量中每一个维度的大小代表每个词对这篇新闻主题的贡献。当新闻从文字变成了数字后，计算机就有可能“算一算”新闻之间是否相似了。

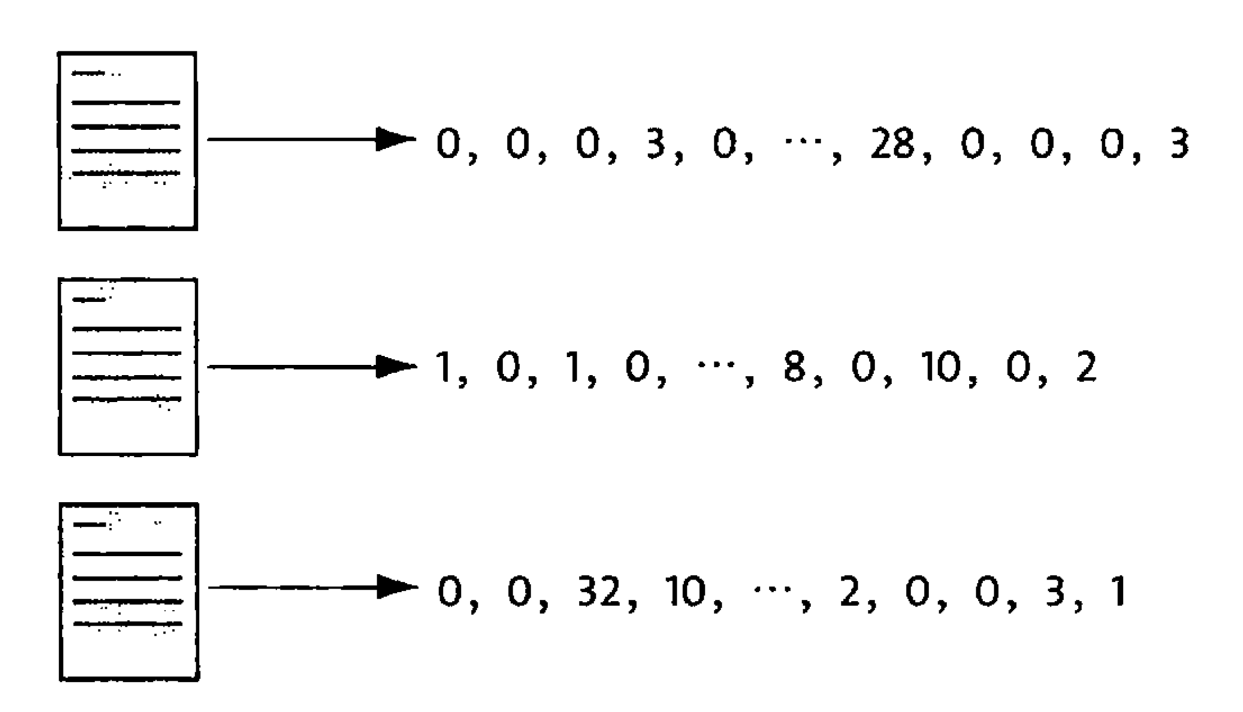


图 14.1 一篇篇文章变成了一串串数字

2 向量距离的度量

世界各国无论是哪门语言的“语文课”（Language Art），老师教授写作时都会强调特定的主题用特定的描述词。几千年来，人类已经形成了这样的写作习惯。因此，同一类新闻一定是某些主题词用得较多，另外一些词则用得少。比如金融类的新闻，这些词出现的频率就很高：**股票，利息，债券，基金，银行，物价，上涨**。而这些词出现的就少：**二氧化碳，宇宙，诗歌，木匠，诺贝尔，包子**。反映在每一篇新闻的特征上，如果两篇新闻属于同一类，它们的特征向量在某几个维度的值都比较大，而在其他维度的值都比较小。反过来看，如果两篇新闻不属于同一类，由于用词的不同，它们的特征向量中，值较大的维度应该没有什么交集。这样就定性地认识到两篇新闻的主题是否接近，取决于它们的特征向量“长得像不像”。当然，我们还需要定量地衡量两个特征向量之间的相似性。

学过向量代数的人都知道，向量实际上是多维空间中从原点出发的有向线段。

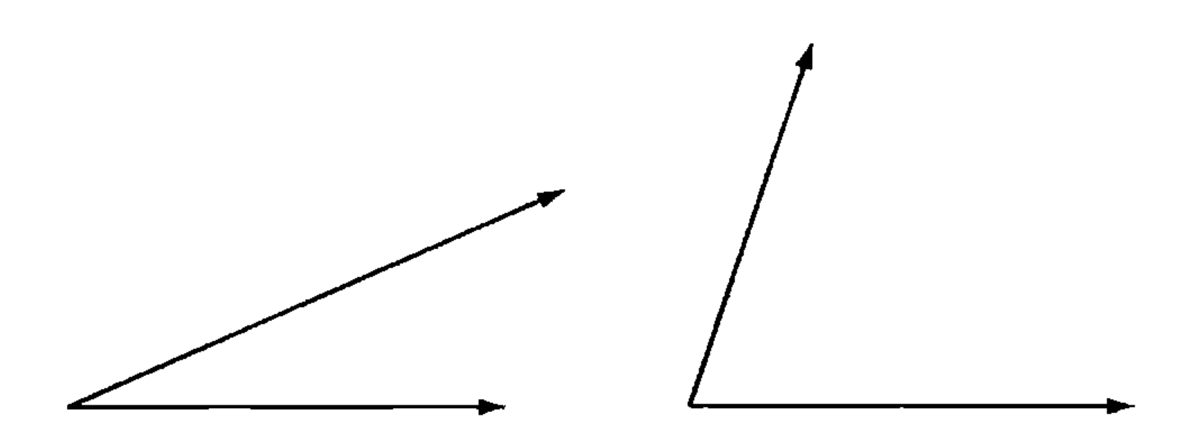


图 14.2 向量的夹角是衡量两个向量相近程度的度量。图中左边的两个向量距离较近，右边两个距离较远

不同的新闻，因为文本长度的不同，它们的特征向量各个维度的数值也不同。一篇 10 000 字的文本，各个维度的数值都比一篇 500 字的文本来得大，因此单纯比较各个维度的大小并没有太大意义。但是，向量的方向却有很大的意义。如果两个向量的方向一致，说明相应的新闻用词的比例基本一致。因此，可以通过计算两个向量的夹角来判断对应的新闻主题的接近程度。而要计算两个向量的夹角，就要用到余弦定理了。比

如上图 14.2 中，左边两个向量的夹角小，距离就较“近”，相反，右边两个向量的夹角大，距离就“远”。

我们对余弦定理都不陌生，它描述了三角形中任何一个夹角和三个边的关系，换句话说，给定三角形的三条边，可以用余弦定理求出三角形各个角的角度。假定三角形的三条边为a ,b和c，对应的三个角为A，B和C。

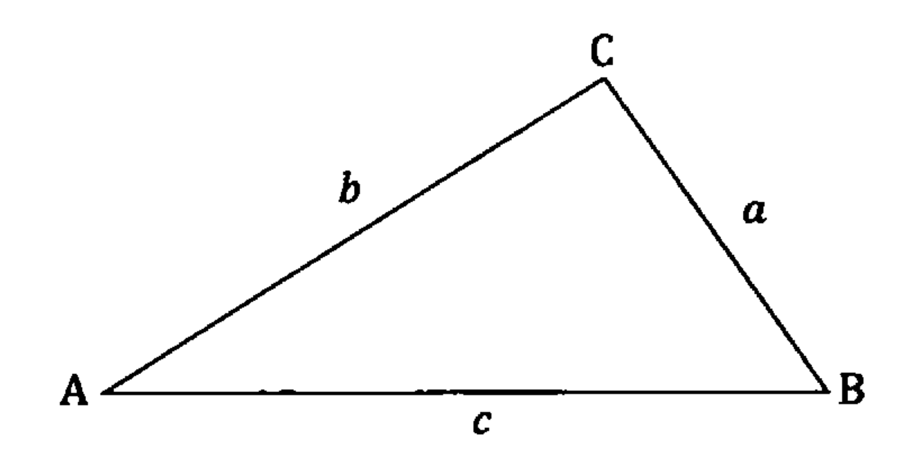


图 14.3 余弦的计算

那么$\angle A$的余弦

$$\cos(A)=\frac{b^2+c^2-a^2}{2bc} \quad (14.1)$$

如果将三角形的两边b和c看成是两个以A为起点的向量，那么上述公式等价于

$$\cos(A)=\frac{<b,c>}{|b|\cdot|c|} \quad (14.2)$$

其中，分母表示两个向量b和c的长度，分子表示两个向量的内积。举一个具体的例子，假如新闻X和新闻Y对应的向量分别是

$x_1, x_2, \ldots, x_{64000}$ 和 $y_1, y_2, \ldots, y_{64000}$，

那么它们夹角的余弦等于

$$\cos\theta=\frac{x_1y_1+x_2y_2+\cdots+x_{64000}y_{64000}}{\sqrt{x_1^2+x_2^2+\cdots x_{64000}^2}\cdot\sqrt{y_1^2+y_2^2+\cdots y_{64000}^2}} \quad (14.3)$$

由于向量中的每一个变量都是正数，因此余弦的取值在 0 和 1 之间，也就是说夹角在 0 度到 90 度之间。当两条新闻向量夹角的余弦等于 1 时，这两个向量的夹角为零，两条新闻完全相同；当夹角的余弦接近于 1 时，两条新闻相似，从而可以归成一类；夹角的余弦越小，夹角越大，两条新闻越不相关。当两个向量正交时（90 度），夹角的余弦为零，说明两篇新闻根本没有相同的主题词，它们毫不相关。

现在把一篇篇文字的新闻变成了按词典顺序组织起来的数字(特征向量)，又有了计算相似性的公式，就可以在此基础上讨论新闻分类的算法了。具体的算法分为两种情形。第一种比较简单，假定我们已知一些新闻类别的特征向量 $x_1, x_2, \cdots x_k$，那么对于任何一个要被分类的新闻 Y，很容易计算出它和各类新闻特征向量的余弦相似性（距离），并且分到它该去的那一类中。至于这些新闻类别的特征向量，既可以手工建立（工作量非常大而且不准确），也可以自动建立（以后会介绍）。第二种情形就比较麻烦了，即如果事先没有这些新闻类别的特征向量怎么办。我在约翰·霍普金斯大学的同学弗洛里安（Radu Florian）[3] 和教授雅让斯基给了一个自底向上不断合并的办法 [4]，大致思想如下：

1. 计算所有新闻之间两两的余弦相似性，把相似性大于一个阈值的新闻合并成一个小类（Subclass）。这样 N 篇新闻就被合并成 N_1 个小类，当然 $N_1 < N$。

2. 把每个小类中所有的新闻作为一个整体，计算小类的特征向量，再计算小类之间两两的余弦相似性，然后合并成大一点的小类，假如有 N_2 个，当然 $N_2 < N_1$。

这样不断做下去，类别越来越少，而每个类越来越大。当某一类太大时，这一类里一些新闻之间的相似性就很小了，这时就要停止上述迭代的过程了。至此，自动分类完成。下图是弗洛里安给出的一个真实的文本分类迭代和聚合的过程，图中左边的每一个点都代表一篇文章，因为数量太多，因此密密麻麻地连成了一片。每一次迭代，子类数量就不断减少，

3 目前是 IBM 华生实验室的科学家。

4 Radu Florian and David Yarowsky, Dynamic nonlocal language modeling via hierarchical topic-based adaptation, ACL 1999

当子类的数量比较少时，就会看清楚这些子类了。

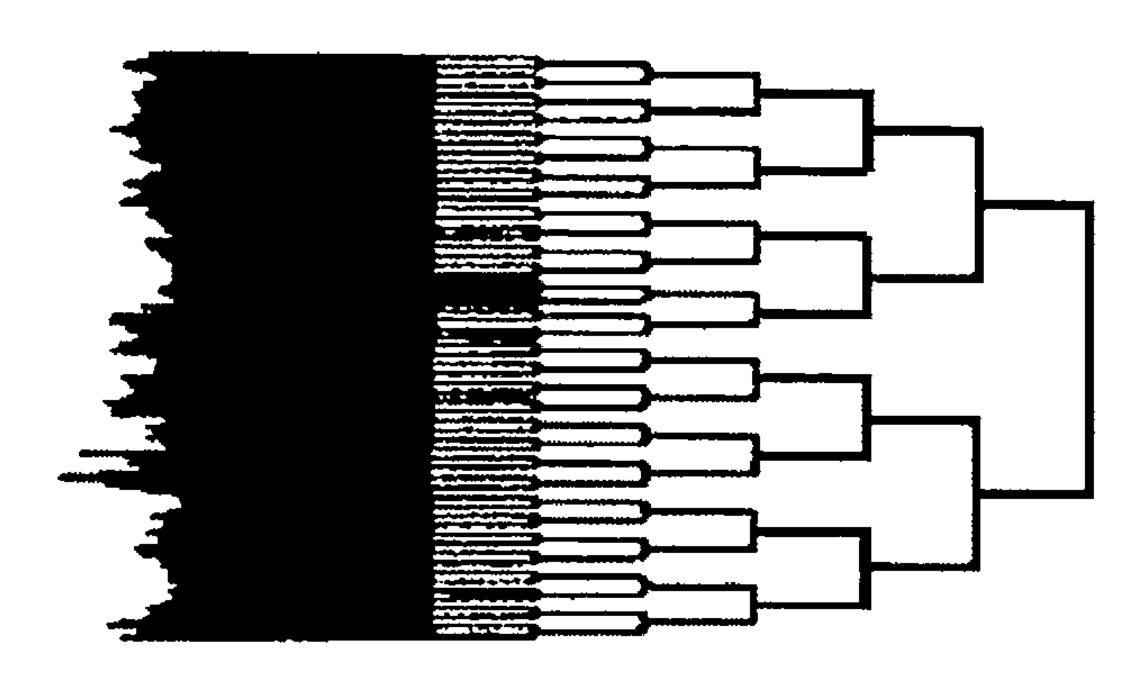


图 14.4　真实的文本分类聚合过程

当然，这里面有很多的技术细节，有兴趣和基础的读者可以读他们的论文。

弗洛里安和雅让斯基 1998 年做这项工作的动机很有意思。当时雅让斯基是某个国际会议的程序委员会主席，需要把提交上来的几百篇论文发给各个专家去评审决定是否录用。为了保证评审的权威性，需要把每个研究方向的论文交给这个方向最有权威的专家。虽然论文的作者自己给论文定了方向，但是范围太广，没有什么指导意义。雅让斯基当然没有时间浏览这近千篇论文，然后再去分发，于是就想了这个将论文自动分类的方法，由他的学生弗洛里安很快实现了。接下来几年的会议都是这么选择评审专家的。从这件事我们可以看出美国人做事的一个习惯：美国人总是倾向于用机器（计算机）代替人工来完成任务。虽然在短期需要做一些额外的工作，但是从长远看可以节省很多时间和成本。

余弦定理就这样通过新闻的特征向量和新闻分类联系在一起。我们在中学学习余弦定理时，恐怕很难想象它可以用来对新闻进行分类。在这里，我们再一次看到数学工具的用途。

3　延伸阅读：计算向量余弦的技巧

读者背景知识：数值分析。

3.1　大数据量时的余弦计算

利用公式（14.2）计算两个向量夹角时，计算量为 $O(|a|+|b|)$，如果假定其中一个向量更长，不失一般性 $|a|>|b|$，这样复杂度为 $O(|a|)$。如果要比较一篇新闻和所有其他 N 篇新闻的相关性，那么计算复杂度为 $O(N\cdot|a|)$。如果要比较所有 N 篇新闻之间两两的相关性，计算复杂度为 $O(N^2\cdot|a|)$。注意，这还只是一次迭代。因此，这个计算量是很大的。我们假定词汇表的大小为 10 万，那么向量的长度也是这么大，假定需要分类的新闻为 10 万篇，总的计算量在 10^{15} 这个数量级。如果用 100 台服务器，每台服务器的计算能力是每秒 1 亿次，那么每次迭代的计算时间在 10 万秒，即大约 1 天。几十次迭代就需要两三个月，这个速度显然很慢。

这里面可简化的地方非常多。首先分母部分（向量的长度）不需要重复计算，计算向量 a 和向量 b 的余弦时，它们的长度可以存起来，等计算向量 a 和向量 c 的余弦时，将 a 的长度直接调出来使用即可。这样，上面的计算量可以节省 2/3，也就是只有公式（14.2）中分子的部分需要两两计算。当然这还没有从根本上降低算法的复杂度。

第二个可以简化的地方就是在计算（14.2）中的分子，即两个向量内积的时候，只需考虑向量中的非零元素即可。那么计算的复杂度取决于两个向量中，非零元素个数的最小值。如果一篇新闻的一般长度不超过 2 000 词，那么非零元素的个数一般也就是 1000 词左右，这样计算的复杂度大约可以下降 100 倍，计算时间从“天”这个量级降到十几分钟这个量级。

第三个简化是删除虚词，这里的虚词包括搜索中的非必留词，诸如“的”、“是”、“和”，以及一些连词、副词和介词，比如“因为”、“所以”、“非常”等等。我们在上一节中分析过，只有同一类新闻，用词才有很大的重复性，

不同类的新闻用词不大相同。因此，去掉这些虚词后，不同类的新闻，即使它们的向量中还有不少非零元素，但是共同的非零元素并不多，因此要做的乘法并不多。大多数情况下，都可以跳过去（因为非零元素乘以零，结果还是零）。这样，计算时间还可以缩短几倍。因此，10 万篇新闻两两比较一下，计算时间也就是几分钟而已。如果做几十次迭代，可以在一天内计算完。

特别需要指出的是，删除虚词后，不仅可以提高计算速度，对新闻分类的准确性也大有好处，因为虚词的权重其实是一种干扰分类正常进行的噪音。这一点，和通信中过滤掉低频噪音是同样的原理。通过这件事，我们也可以看出自然语言处理和通信的很多道理是相通的。

3.2　位置的加权

和计算搜索相关性一样，出现在文本不同位置的词在分类时的重要性也不相同。显然，出现在标题中的词对主题的贡献远比出现在新闻正文中的重要。而即使在正文中，出现在文章开头和结尾的词比出现在中间的词重要。在中学学习语文和大学学习英语文学时，老师都会强调这一点——阅读时要特别关注第一段和最后一段，以及每个段落的第一个句子。在自然语言处理中这个规律依然有用。因此，要对标题和重要位置的词进行额外的加权，以提高文本分类的准确性。

4　小结

本章介绍的这种新闻归类的方法，准确性很好，适用于被分类的文本集合在百万这个数量级。如果大到亿这个数量级，那么计算时间还是比较长的。对于更大规模的文本处理，我们将在下一章中介绍一种更快速，但是相对粗糙一些的方法。

第15章 矩阵运算和文本处理中的两个分类问题

我在大学学习线性代数时，实在想不出这门课除了告诉我们如何解线性方程外，还能有什么别的用途。关于矩阵的许多概念，比如特征值等，更是脱离日常生活。后来在数值分析中又学了很多矩阵的近似算法，还是看不到可以应用的地方。当时选这些课，完全是为了挣学分得学位，今天大部分大学生恐怕也是如此。我想，很多同学都多多少少有过类似的经历。直到后来长期做自然语言处理的研究，我才发现数学家们提出的那些矩阵的概念和算法，是有实际应用的意义的。

1 文本和词汇的矩阵

在自然语言处理中，最常见的两个分类问题分别是，将文本按主题归类（比如将所有介绍奥运会的新闻归到体育类）和将词汇表中的字词按意思归类（比如将各种运动的项目名称都归成体育一类）。这两个分类问题都可以通过矩阵运算来圆满地、一次性地解决。为了说明如何用矩阵这个工具来解决这两个问题，让我们来看一看前一章中介绍的余弦定理和新闻分类的本质。

新闻分类乃至各种分类其实是一个聚类问题，关键是计算两篇新闻的相似程度。为了完成这个过程，我们要将新闻变成代表它们内容的实词，

然后再变成一组数，具体说是向量，最后求这两个向量的夹角。当这两个向量夹角很小时，新闻就相关；当它们垂直或者说正交时，新闻则无关。从理论上讲，这种算法非常漂亮。但是因为要对所有新闻做两两计算，而且要进行很多次迭代，因此耗时特别长，尤其是当新闻的数量很大，同时词表也很大的时候。我们希望有一个办法，一次就能把所有新闻相关性计算出来。这个一步到位的办法是利用矩阵运算中的奇异值分解（Singular Value Decomposition，简称 SVD）。

现在让我们来看看奇异值分解是怎么回事。首先，需要用一个大矩阵A来描述成千上万篇文章和几十上百万词的关联性。在这个矩阵中，每一行对应一篇文章，每一列对应一个词，如果有N个词，M篇文章，那么就是一个如下所示的$M \times N$的矩阵。

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1N} \\ \cdots & & & & \cdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{iN} \\ \cdots & & & & \cdots \\ a_{M1} & \cdots & a_{Mj} & \cdots & a_{MN} \end{bmatrix} \quad (15.1)$$

其中，第i行、第j列的元素a_{ij}，是字典中第j个词在第i篇文章中出现的加权词频（比如用词的 TF-IDF 值）。读者也许能猜到，这个矩阵会非常非常大，比如$M = 1\,000\,000$，$N = 500\,000$，100 万乘以 50万，即 5 000 亿个元素，如果以 5 号字体打印出来，有两个西湖那么大！

奇异值分解，就是把上面这样一个大矩阵，分解成三个小矩阵相乘，如图 15.1 所示。比如把上面例子的矩阵分解成一个 100 万乘以 100 的矩阵X，一个 100 乘以 100 的矩阵B，和一个 100 乘以 50 万的矩阵Y。这三个矩阵的元素总数加起来也不过 1.5 亿，不到原来的三千分之一。相应的存储量和计算量都会小三个数量级以上。

$$\underset{1\,000\,000\times 500\,000}{\boxed{A}} = \underset{1\,000\,000\times 100}{\boxed{X}}\ \underset{100\times 100}{\boxed{B}}\ \underset{100\times 500\,000}{\boxed{Y}}$$

图 15.1 大矩阵分解成三个小矩阵

三个矩阵有非常清晰的物理含义。第一个矩阵 X 是对词进行分类的一个结果。它的每一行表示一个词，每一列表示一个语义相近的词类，或者简称为语义类。这一行的每个非零元素表示这个词在每个语义类中的重要性（或者说相关性），数值越大越相关。举一个 4×2 的小矩阵的例子来说明。

$$X = \begin{bmatrix} 0.7 & 0.15 \\ 0.22 & 0.49 \\ 0 & 0.92 \\ 0.3 & 0.03 \end{bmatrix} \tag{15.2}$$

这里面有四个词，两个语义类，第一个词和第一个语义类比较相关（相关性 0.7），和第二个语义类不太相关（相关性 0.15）。第二个词正好相反。第三个词只和第二个语义类相关，和第一个完全无关。第四个词和每一类都不太相关，因为它对应的两个元素 0.3 和 0.03 都不大，但是相比较而言，和第一类相对相关，而第二类基本无关。

最后一个矩阵 Y 是对文本的分类结果。它的每一列对应一个文本，每一行对应一个主题。这一列中每个元素表示这篇文本在不同主题中的相关性。我们同样用一个 2×4 的小矩阵来说明。

$$Y = \begin{bmatrix} 0.7 & 0.15 & 0.22 & 0.39 \\ 0 & 0.92 & 0.08 & 0.53 \end{bmatrix} \tag{15.3}$$

这里面有四篇文本，两个主题。第一篇文本很明显，属于第一个主题。第二篇文本和第二个主题非常相关（相似性 0.92），而和第一个有一点

点关系（0.15）。第三篇文本和两个主题都不很相关，相比较而言，和第一个主题的关系稍微近一点。第四篇文本，和两个主题都有一定的相关性，不过和第二个主题更接近一些。如果每一列只保留最大值，其余的都改为零，那么每一篇文本都被唯一地分到一类主题中，即第一、三篇文本属于第一个主题，第二、四篇属于第二个主题。这个结果和我们在上一章利用余弦定理做分类得到的情况类似 —— 每篇文本都被分到了一个主题。

中间的矩阵则表示词的类和文章的类之间的相关性。我们用下面 2×2 的矩阵来说明。

$$B = \begin{bmatrix} 0.7 & 0.21 \\ 0.18 & 0.63 \end{bmatrix} \qquad (15.4)$$

在这个矩阵 B 中，第一个词的语义类和第一个主题相关，和第二个主题没有太多关系。而第二个词的语义类则相反。

因此，只要对关联矩阵 A 进行一次奇异值分解，就可以同时完成近义词分类和文章的分类。同时，还能得到每个主题和每个词的语义类之间的相关性。这个结果非常漂亮！

现在剩下的唯一问题，就是如何用计算机进行奇异值分解。这时，线性代数中的许多概念，比如矩阵的特征值，以及数值分析的各种算法等等就统统派上用场了。对于不大的矩阵，比如几万乘以几万的矩阵，用计算机上的数学工具 MATLAB 就可以进行了。但是更大的矩阵，比如上百万乘以上百万，奇异值分解的计算量非常大，就需要很多台计算机并行处理了。虽然 Google 早就有了 MapReduce 等并行计算的工具，但是由于奇异值分解很难拆成不相关子运算，即使在 Google 内部以前也无法利用并行计算的优势来分解矩阵。直到 2007 年，Google 中国（谷歌）的张智威博士带领几个中国的工程师及实习生实现了奇异值分解的并行算法，这是 Google 中国对世界的一个贡献。

2　延伸阅读：奇异值分解的方法和应用场景

读者背景知识：线性代数。

在这一节里，我们大致介绍一下奇异值分解的算法。在严格数学意义上的奇异值分解是这样定义的，矩阵 A 可以如下分解成三个矩阵的乘积。

$$A_{MN} = X_{MM} \times B_{MN} \times Y_{NN} \tag{15.5}$$

其中 X 是一个酉矩阵（Unitary Matrix），Y 则是一个酉矩阵的共轭矩阵。酉矩阵的定义是它和它的共轭矩阵转置相乘等于单位阵，因此，它们都是方形的矩阵。而 B 是一个对角阵，即只有对角线上是非零值。维基百科给出了这样一个实例：

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{bmatrix} \tag{15.6}$$

$$X = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$Y = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sqrt{0.2} & 0 & 0 & 0 & \sqrt{0.8} \\ 0 & 0 & 0 & 1 & 0 \\ -\sqrt{0.8} & 0 & 0 & 0 & \sqrt{0.2} \end{bmatrix} \tag{15.7}$$

很容易验证 X 和 Y 都是酉矩阵。

从这个公式来看，未作近似的奇异值分解没有像上一节中展示的那样降低矩阵的维度。但是由于对角矩阵 B 对角线上的元素很多值相对其他的非常小，或者干脆为零，可以省略。因此，奇异值分解后，一个超大矩阵就变成我们上一节中三个小矩阵的乘积。

1
除了两行对角线元素非零，剩下的都是零。

奇异值分解一般分两步进行。首先，将矩阵A变换成一个双对角矩阵[1]，这个过程的计算量是$O(MN^2)$，当然这里假设$M > N$，否则就是$O(NM^2)$。当然，我们依然可以利用矩阵A的稀疏性大大缩短计算时间。第二步是将双对角矩阵变成奇异值分解的三个矩阵。这一步计算量只是第一步的零头，可以忽略不计。

在文本分类中，M对应文本的数量，N对应词典大小。如果比较奇异值分解的计算复杂度和利用余弦定理计算文本相似度（一次迭代）的时间，它们处于同一个数量级，但是前者不需要多次迭代，因此计算时间短不少。奇异值分解的另一个大问题是存储量较大，因为整个矩阵都需要存在内存里，而利用余弦定理的聚类则不需要。

3 小结

相比上一章介绍的利用文本特征向量余弦的距离自底向上的分类方法，奇异值分解的优点是能较快地得到结果（在实际应用中），因为它不需要一次次地迭代。但是用这种方法得到的分类结果略显粗糙，因此，它适合处理超大规模文本的粗分类。在实际工作中，可以先进行奇异值分解，得到粗分类结果，再利用计算向量余弦的方法，在粗分类结果的基础上，进行几次迭代，得到比较精确的结果。这样，将这两个方法先后使用，可以充分利用这两个方法的优势，既节省时间，又获得很好的准确性。

参考文献：

1. 1. J. R. Bellegarda, Exploiting latent semantic information in statistical language modeling, Proceedings of the IEEE, Volume:88 Issue:8, 1279-1296, August 2008.

第16章 信息指纹及其应用

1 信息指纹

在前面的章节中，我们讲到，一段文字所包含的信息，就是它的信息熵。如果对这段信息进行无损压缩编码，理论上编码后的最短长度就是它的信息熵。当然，实际编码长度总是要略长于它的信息熵的比特数。但是，如果仅仅要区分两段文字或者图片，则远不需要那么长的编码。任何一段信息（包括文字、语音、视频、图片等等），都可以对应一个不太长的随机数，作为区别它和其他信息的指纹（Fingerprint）。只要算法设计得好，任何两段信息的指纹都很难重复，就如同人类的指纹一样。信息指纹在加密、信息压缩和处理中有着广泛的应用。

我们在“图论和网络爬虫”一章中提到，为了防止重复下载同一个网页，需要在哈希表中记录已经访问过的网址（URL）。但是在哈希表中以字符串的形式直接存储网址，既费内存空间，又浪费查找时间。现在的网址一般都较长，比如，如果在Google、搜搜或者百度上查找“吴军 数学之美”，对应的网址长度在100个字符以上。下面是百度的链接：

http://www.baidu.com/s?ie=gb2312&bs=%CA%FD%D1%A7%D6%AE%C3%C0&sr=&z=&cl=3&f=8&wd=%CE%E2%BE%FC+%CA%FD%D1%A7%D6%AE%C3%C0&ct=0

到了 2010 年，互联网上的网页总数在 5 000 亿这个量级，假定网址的平均长度为 100 个字符，那么存储 5 000 亿个网址本身至少需要 50TB，即 5 000 万兆字节的容量。考虑到哈希表的存储效率一般只有 50% 左右，故实际需要的内存在 100TB 以上。如果一台服务器内存是 50GB（2010 年的水平），也需要 2 000 台服务器来存放这些内容。另外，即使有这么多服务器，将这些网址放到了计算机的内存中，由于网址长度不固定，以字符串的形式查找，效率会很低。

因此，如果能够找到一个函数，将这 5 000 亿个网址随机地映射到 128 位二进制（即 128 比特），也就是 16 个字节的整数空间，比如将上面那个很长的字符串对应成一个如下的随机数：

893249432984398432980545454543

这样，每个网址就只需要占用 16 个字节而不是原来的 100 个。这就能把存储网址的内存需求量降至原来的 1/6 不到。这个 16 个字节的随机数，就称做该网址的信息指纹。可以证明，只要产生随机数的算法足够好，就能保证几乎不可能有两个字符串的指纹相同，就如同不可能有两个人的指纹相同一样。由于指纹是固定的 128 位二进制整数，因此，查找的计算量与字符串相比，要小得多。网络爬虫在下载网页时，它将访问过的网页地址都变成一个个信息指纹，存到哈希表中，每当遇到一个新网址，计算机就计算出它的指纹，然后比较该指纹是否已经在哈希表中，来决定是否下载这个网页。这种整数的查找比原来的字符串查找快几十倍。对于要求不是很高的网络爬虫，甚至采用 64 位二进制就足够了，这样可以进一步节省存储空间和运算时间。

上面那个网址（字符串）的信息指纹的计算方法一般分为两步。首先，将这个字符串看成是一个特殊的、长度很长的整数。这一步非常容易，因为所有的字符在计算机里都是按照整数来存储的。接下来就需要一个产生信息指纹的关键算法：伪随机数产生器算法（Pseudo-Random Number Generator，简称 PRNG），通过它将任意很长的整数转换成特

定长度的伪随机数。最早的 PRNG 算法是由计算机之父冯·诺伊曼提出来的。他的办法非常简单，就是将一个数的平方掐头去尾，取中间的几位数。比如一个四位的二进制数 1001（相当于十进制的 9），其平方为 01010001（十进制的 81），掐头去尾，剩下中间的 4 位 0100。当然这种方法产生的数字并不很随机，也就是说，两个不同信息很有可能有同一指纹，比如 0100（十进制的 4），它按照这个方法产生的指纹也是 0100。现在常用的梅森旋转算法（Mersenne Twister）要好得多。

信息指纹的用途远不止网址的消重，它的孪生兄弟是密码。信息指纹的一个特征是其不可逆性，也就是说，无法根据信息指纹推出原有信息。这种性质，正是网络加密传输所需要的。比如说，一个网站可以根据用户本地客户端的 cookie 识别不同用户，这个 cookie 就是一种信息指纹。但是网站无法根据信息指纹了解用户的身份，这样就可以保护用户的隐私。但是 cookie 本身并没有加密，因此通过分析 cookie 还是能知道某台计算机访问了哪些网站。为了保障信息安全，一些网站（比如银行的）采用加密的 HTTPS，用户访问这些网站留下的 cookie 本身也需要加密。加密的可靠性，取决于是否很难人为地找到拥有同一指纹的信息，比如一个黑客是否能产生出某位用户的 cookie。从加密的角度讲梅森旋转算法还不够好，因为它产生的随机数还有一定的相关性，破解一个就等于破解了一大批。

在互联网上加密要使用基于加密的伪随机数产生器（Cryptographically Secure Pseudo-Random Number Generator，简称 CSPRNG）。常用的算法有 MD5 或者 SHA-1 等标准，它们可以将不定长的信息变成定长的 128 位或者 160 位二进制随机数。值得一提的是，SHA-1 以前被认为是没有漏洞的，现在已经被中国的王小云教授证明存在漏洞。但是大家不必恐慌，因为这和黑客能真正攻破你的注册信息还是两回事。

2 信息指纹的用途

信息指纹的历史虽然很悠久，但真正的广泛应用是在有了互联网以后，近十年才渐渐热门起来。

上面一节讲述了在网络爬虫中利用信息指纹可以快速而经济（节省服务器）地判断一个网页是否已下载过。信息指纹在互联网和自然语言处理中还有非常多的应用，这里不可能（也不必要）一一列举，只是找几个有代表性的例子。

2.1 判定集合相同

在网页搜索中，有时需要判断两个查询用词是否完全相同（但是次序可能不同），比如“北京　中关村　星巴克”和“星巴克　北京　中关村”用词完全相同。更普遍的讲法是判断两个集合是否相同（比如一个人是否用两个不同的 Email 帐号对同一群人群发垃圾邮件）。解决这个问题有各种各样的方法，没有绝对正确的和错误的，但是有好的方法和笨的方法。

最直接的笨办法是对这个集合中的元素一一做比较，这个方法计算的时间复杂度是 $O(N^2)$，其中 N 是集合的大小。如果谁面试时这么回答我，我肯定不会让他通过。

稍微好一点的办法是将两个集合的元素分别排序，然后顺序比较，这样计算时间的复杂度是 $O(N\log N)$，比前面那种方法好了不少，但还不是很好。与这个方法处于同一个水平的是将第一个集合放在一张哈希表中，然后把第二个集合的元素一一和哈希表中的元素作对比。这个方法的时间复杂度为 $O(N)$，达到了最佳[1]。但是额外使用了 $O(N)$ 的空间，而且代码很复杂，不完美。

1 $O(N)$ 的时间复杂度是不可能突破的，因为毕竟要扫描一遍所有 N 个元素。

完美的方法是计算这两个集合的指纹，然后直接进行比较。我们定义一

个集合 $S=\{e_1,e_2,\cdots,e_n\}$ 的指纹 $FP(S)=FP(e_1)+FP(e_2)+\cdots+FP$，其中 $FP(e_1)$，$FP(e_2)$，…，$FP(e_n)$ 分别为 S 中这些元素对应的指纹。由于加法的交换率，保证集合的指纹不因元素出现的次序而改变，如果两个集合元素相同，那么它们的指纹一定相同。当然，不同元素的指纹也相同的概率非常非常小，在工程上完全可以忽略。我们在延伸阅读里会告诉读者这个概率有多小。

利用信息指纹的方法计算的复杂度是 $O(N)$，而且不需要额外的空间，因此是最佳方法。

类似的应用还有很多，如比较网上的一首歌是否是盗版别人的，只要算一算这两个音频文件的信息指纹即可。

2.2　判定集合基本相同

爱思考的读者可能会挑战我：发垃圾邮件的人哪有这么傻，从两个帐号发出的垃圾邮件收信人都相同？如果稍微变上一两个，上面的方法不就不起作用了吗？解决这个问题需要我们能够快速判断两个集合是否基本相同，其实只要将上面的方法稍作改动即可。

可以分别从两个帐号群发的接收电子邮件地址清单（Email Address List）中按照同样的规则随机挑选几个电子邮件的地址，比如尾数为 24 的。如果它们的指纹相同，那么很有可能这两个接收的电子邮件单子基本相同。由于挑选的数量有限，通常是个位数，因此也很容易判断是否是 80%，或者 90% 重复。

上述判断集合基本相同的算法有很多实际的应用，比如在网页搜索中，判断两个网页是否是重复的。如果把两个网页（的正文）从头比到尾，计算时间太长，也没有必要。我们只需对每个网页挑出几个词，这些词构成网页的特征词集合。然后计算和比较这些特征集合的信息指纹即可。在两个被比较的网页中，常见的词一般都会出现，不能作为这两篇文章的

特征。只出现一次的词，很可能是噪音，也不能考虑。在剩下的词中，我们知道 IDF 大的词鉴别能力强，因此只需找出每个网页中 IDF 最大的几个词，并且算出它们的信息指纹即可。如果两个网页这么计算出来的信息指纹相同，它们基本上是相同的网页。为了允许有一定的容错能力，在 Google 里采用了一种特定的信息指纹 —— 相似哈希（Simhash）。相似哈希的原理会在延伸阅读中介绍。

上面的算法稍作改进还可以判断一篇文章是否抄袭另一篇文章。具体的做法是，将每一篇文章切成小的片段，然后对这些片段用上述方法选择特征词的集合，并且计算它们的指纹。只要比较这些指纹，就能找出大段相同的文字，最后根据时间先后，找到原创的和抄袭的。Google 实验室利用这个原理做了一个名为 CopyCat 的项目，能够很准确地找出原文和转载（拷贝）的文章。

2.3 YouTube 的反盗版

Google 旗下的 YouTube 是全球最大的视频网站，和国内的视频网站不同，YouTube 自身不提供和上传任何内容，完全由用户自己提供。这里的用户既包括专业的媒体公司，比如 NBC 和迪士尼，也包括个人用户。由于对后者没有太多上传视频的限制，一些人会上传专业媒体公司的内容。这件事不解决就会动摇 YouTube 的生存基础。

从上百万视频中找出一个视频是否是另一个视频的盗版，并不是一件容易的事情。一段几分钟的视频，文件大小有几兆到几十兆，而且还是压缩的，如果恢复到每秒 30 帧的图像， 数据量就会大得不得了。因此，没有人通过直接比较两段视频的方法来确定它们是否相似。

视频的匹配有两个核心技术，关键帧的提取和特征的提取。MPEG 视频（在 NTSC 制的显示器上播放）虽然每秒钟有 30 帧图像，但是每一帧之间的差异不大。（否则我们看起来就不连贯了。）只有极少数的帧是完整的图像，这些称为关键帧。其余帧存储的只是和关键帧相比的差异值。关键帧对

于视频的重要性，就如同主题词对于新闻的重要性一样。因此，处理视频图像首先是找到关键帧，接下来就是要用一组信息指纹来表示这些关键帧了。

有了这些信息指纹后，接下来查盗版的事情就类似于比较两个集合元素是否相同了。Google 收购 YouTube 后，由 Google 研究院研究图像处理的科学家们开发出的反盗版系统，效果非常好。由于可以找出相同的视频的原创和拷贝，Google 制定了一个很有意思的广告分成策略：虽然所有的视频都可以插入广告，但是广告的收益全部提供给原创的视频，即使广告是插入在拷贝的视频中。这样一来，所有拷贝和上传别人的视频的网站就不可能获得收入。没有了经济利益，也就少了很多盗版和拷贝。

3 延伸阅读：信息指纹的重复性和相似哈希

读者背景知识：概率论、组合数学。

3.1 信息指纹重复的可能性

信息指纹是通过伪随机数产生的。既然是伪随机数，两个不同的信息就有可能产生同样的指纹。这种可能性从理论上讲确实存在，但是非常小。至于有多小，我们就在这一节中分析。

假定一个伪随机数产生的范围是 $0 \sim N-1$，共 N 个。如果是 128 位的二进制，$N = 2^{128}$，这是一个非常巨大的数字。如果随意挑选两个指纹，那么它们重复的可能性就是 $1/N$，不重复的可能性是 $\frac{N-1}{N}$，因为第一个可以是任一个，第二个只有 $N-1$ 的可选余地。如果随意挑选三个指纹，要保证不重复，第三个只有 $N-2$ 的可选余地，因此，三个不重复的概率为 $\frac{(N-1)(N-2)}{N^2}$。以此类推，k 个指纹不重复的概率

$$P_k = \frac{(N-1)(N-2)\dots(N-k+1)}{N^{k-1}}$$ 。

P_k 随着 k 增加而减小，即产生的指纹多到一定程度，就可能有重复的了。如果 $P_k < 0.5$，那么，k 个指纹重复一次的数学期望超过 1。现在来估计一下这时候 k 的最大值。

上述不等式等价于

$$P_{k+1} = \frac{(N-1)(N-2)\dots(N-k)}{N^k} < 0.5 \qquad (16.1)$$

当 $N \to \infty$，

$$P_{k+1} \approx e^{-\frac{1}{n}}e^{-\frac{2}{n}}\dots e^{-\frac{k}{n}} = \exp\left(-\frac{k(k+1)}{2N}\right) \qquad (16.2)$$

这个概率需要小于 0.5，因此

$$P_{k+1} \approx \exp\left(-\frac{k(k+1)}{2N}\right) < 0.5 \qquad (16.3)$$

这等效于

$$k^2 + k + 2N\ln 0.5 > 0.5 \qquad (16.4)$$

由于 $k > 0$，上述不等式有唯一解

$$k > \frac{-1+\sqrt{1+8N\log 2}}{2} \qquad (16.5)$$

也就是说，对于一个很大的 N，k 是一个很大的数字。如果用 MD5 指纹（虽然它有缺陷），它有 128 位二进制，$k > 2^{64} \approx 1.8\times 10^{19}$。也就是说，每一千八百亿亿次才能重复一次，因此，不同信息产生相同指纹的可能性几乎为零。即使采用 64 位的指纹，重复的可能性依然很低。

3.2 相似哈希（Simhash）

相似哈希是一种特殊的信息指纹，是 Moses Charikar 在 2002 年提出来的[2]，但是真正得到重视是当 Google 在网页爬虫中使用它，并且把结果发表在 WWW 会议上以后[3]。虽然 Charikar 的论文写得比较晦涩，但是相

2 Moses Charikar, Similarity Estimation Techniques from Rounding Algorithms, Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002.

3 Gurmeet Singh Manku, Arvind Jain and Anish Das Sarma, Detecting Near-Duplicates for Web Crawling, WWW2007, 2007.

似哈希的原理其实并不复杂。不妨用一个 Google 在下载网页时排查重复网页的例子来说明。

假定一个网页中有若干的词 $t_1, t_2, \ldots, t_k$，它们的权重（比如 TF-IDF 值）分布为 $w_1, w_2, \ldots, w_k$。先计算出这些词的信息指纹，为便于说明，假定只用 8 位二进制的信息指纹。当然在实际工作中不能用这么短的，因为重复度太高。计算相似哈希分为两步。

第一步我称之为扩展，就是将 8 位二进制的指纹扩展成 8 个实数，用 $r_1, r_2, \ldots, r_8$ 表示，这些实数的值如下确定：

首先，将它们的初值设置为 0，然后，看 t_1 的指纹（8 位），如果 t_1 的第 i 位为 1，在 r_i 上加上 w_1；如果为 0，在 r_i 上减去 w_1。例如，t_1 的指纹为 10100110（随便给的），这样处理完 t_1 后，r_1 到 r_8 的值如下：

表 16.1 处理了第一个词后，r_1 到 r_8 的值

$r_1 = 1$	w_1
$r_2 = 0$	$-w_1$
$r_3 = 1$	w_1
$r_4 = 0$	$-w_1$
$r_5 = 0$	$-w_1$
$r_6 = 1$	w_1
$r_7 = 1$	w_1
$r_8 = 0$	$-w_1$

接下来处理第二个词 t_2，加入它的指纹是 00011001，那么根据上面逢 1 相加、逢 0 相减的原则，因为第 1 位是 0，因此 r_1 上应该减去 t_2 的权重 w_2，这样 $r_1 = w_1 - w_2$，如此 $r_2, \ldots, r_8$ 做同样处理，结果如表 16.2。

表 16.2 处理了第一、二个词后，r_1 到 r_8 的值

r_1	$w_1 - w_2$
r_2	$-w_1 - w_2$

续表

r_3	$w_1 - w_2$
r_4	$-w_1 - w_2$
r_5	$-w_1 - w_2$
r_6	$w_1 - w_{21}$
r_7	$w_1 - w_2$
r_8	$-w_1 + w$

当扫描完全部词时，就得到了最后的 8 个数 $r_1, \ldots, r_8$，第一步扩展过程到此结束。假定 $r_1, r_2, \ldots, r_8$ 的值在扩展后变为如表 16.3 所示的那样：

表 16.3　处理完所有的词后，r_1 到 r_8 的值，然后把正数变成 1，负数变成 0

r_1	-.052	0
r_2	-1.2	0
r_3	0.33	1
r_4	0.21	1
r_5	-0.91	0
r_6	-1.1	0
r_7	-0.85	0
r_8	0.52	1

第二步我称之为收缩，就是把 8 个实数变回成一个 8 位的二进制。这个过程非常简单，如果 $r_i > 0$，就把相应的二进制的第 i 位设置成 1，否则设置成 0。这样就得到了这篇文章的 8 位相似哈希指纹。对于上面的例子，这篇文章的 Simhash = 00110001。

相似哈希的特点是，如果两个网页的相似哈希相差越小，这两个网页的相似性越高。如果两个网页相同，它们的相似哈希肯定相同。如果它们只有少数权重小的词不同，其余的都相同，几乎可以肯定它们的相似哈希也会相同。值得一提的是，如果两个网页的相似哈希不同，但是相差很小，则对应的网页也非常相似。用 64 位的相似哈希做对比时，如果只相差一两位，那么对应网页内容重复的可能性大于 80%。这样通过记录

每个网页的相似哈希，然后判断一个新网页的相似哈希是否已经出现过，可以找到内容重复的网页，就不必重复建索引浪费计算机资源了。

信息指纹的原理简单，使用方便，因此用途非常广泛，是当今海量数据处理必不可少的工具。

4 小结

信息指纹可以理解成将一段信息（文字、图片、音频、视频等）随机地映射到一个多维二进制空间中的一个点（一个二进制数字）。只要这个随机函数做得好，那么不同信息对应的这些点不会重合，因此这些二进制的数字就成了原来信息所具有的独一无二的指纹。

参考文献：

1. Moses Charikar, Similarity Estimation Techniques from Rounding Algorithms, Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002.
2. Gurmeet Singh Manku, Arvind Jain and Anish Das Sarma, Detecting Near-Duplicates for Web Crawling, WWW2007, 2007.

第17章　由电视剧《暗算》所想到的——谈谈密码学的数学原理

2007年，我看了电视剧《暗算》，很喜欢它的构思和里面的表演。其中有一个故事提到了密码学，故事本身不错，但是有点故弄玄虚。不过有一点是对的，就是当今的密码学是以数学为基础的。（没有看过《暗算》的读者可以看一下网上的介绍，因为后面会多次提到这部电视剧。）

1　密码学的自发时代

密码学的历史大致可以追溯到两千年前，相传古罗马名将恺撒（Julius Caesar）为了防止敌方截获情报，用密码传送情报。恺撒的做法很简单，就是对二十几个罗马字母建立一张对应表，如表17.1所示。

表17.1　恺撒大帝的明码密码对应表

明码	密码
A	B
B	E
C	A
D	F
E	K
…	…
R	P
S	T
…	…

这样，如果不知道密码本，即使截获一段信息也看不懂，比如收到一个的消息是 ABKTBP，那么在敌人看来是毫无意义的字，通过密码本破解出来就是 CAESAR 一词，即恺撒的名字。这种编码方法史称恺撒大帝，现在市场上还有这一类的玩具卖，见图 17.1。



图 17.1 市场上卖的名为“恺撒大帝”的玩具

当然，学过信息论的人都知道，只要多截获一些情报（即使是加密的），统计一下字母的频率，就可以破解出这种密码。柯南·道尔（Sir Arthur Ignatius Conan Doyle）在他的《福尔摩斯探案集》中“跳舞的小人”的故事里已经介绍了这种小技巧（见图 17.2）。近年来在很多谍报题材的电视剧中，编剧还在经常使用这种蹩脚的密码，比如用菜价（一组数字）传递信息，这些数字对应康熙字典的页码和字的次序。对于学过信息论的人来说，破译这种密码根本不需要密码本，只要多收集几次情报就可以破译出来。

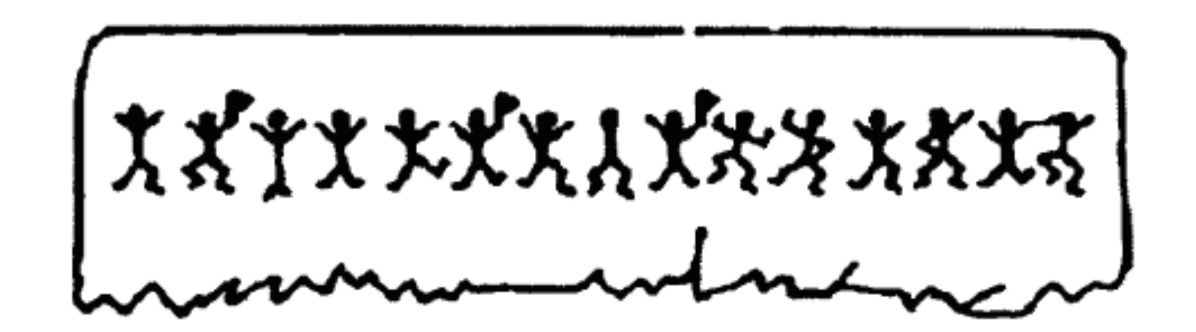


图 17.2 跳舞的小人：看上去很神秘，但是很容易被破解

从恺撒大帝到 20 世纪初很长的时间里，密码的设计者们在非常缓慢地改进技术，因为他们的工作基本上靠经验，没有自觉地应用数学原理（当然当时还没有信息论）。人们渐渐意识到一个好的编码方法会使得解密

者无法从密码中统计出明码的统计信息。有经验的编码者会把常用的词对应成多个密码，使得破译者很难统计出任何规律。比如，如果将汉语中的“是”一词对应于唯一一个编码 0543，那么破译者就会发现 0543 出现的特别多。但如果将它对应成 0543、373、2947 等等 10 个密码，每次随机地选用一个，每个密码出现的次数就不会太多，而且破译者也无从知道这些密码其实对应一个字。这里面已经包含着朴素的概率论的原理。

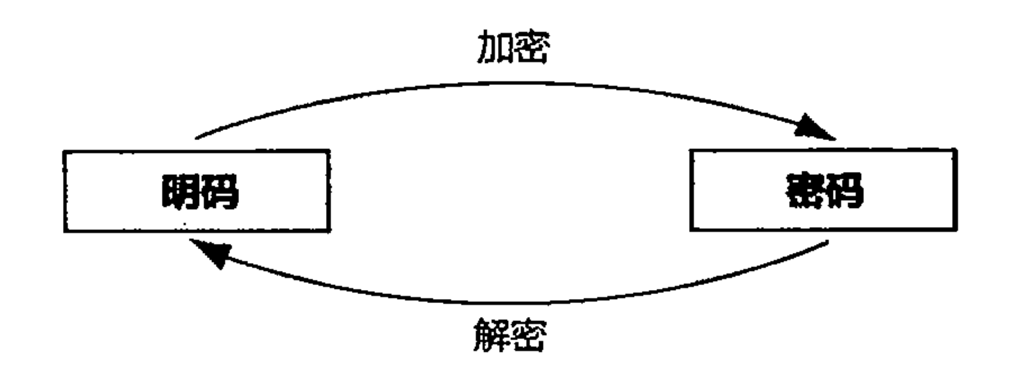


图 17.3　加密和解密是一对函数和反函数

好的密码必须做到不能根据已知的明文和密文的对应推断出新的密文的内容。从数学的角度上讲，加密的过程可以看作是一个函数的运算 F，解密的过程是反函数的运算。明码是自变量，密码是函数值。好的（加密）函数不应该通过几个自变量和函数值就能推出函数。这一点在第二次世界大战前做得很不好。历史上有很多在这方面设计得不周到的密码的例子。比如在第二次世界大战中，日本军方的密码设计就很成问题。美军破获了日本很多密码。在中途岛海战前，美军截获的日军密电经常出现 AF 这样一个地名，应该是太平洋的某个岛屿，但是美军无从知道是哪个。于是，美军就逐个发布自己控制的岛屿有关的假新闻。当美军发出“中途岛供水系统坏了”这条假新闻后，从截获的日军情报中又看到含有 AF 的电文（日军情报内容是 AF 供水出了问题），美军就断定中途岛就是 AF。事实证明判断正确，美军在那里成功地伏击了日本联合舰队。

已故的美国情报专家雅德利（Herbert Osborne Yardley，1889-1958）二战时曾经在重庆帮助中国政府破解日本的密码。他在重庆的两年里做得最成功的一件事，就是破解了日军和重庆间谍的通信密码，并因此破译了几千份日军和间谍之间通信的电文，从而破获了国民党内奸“独臂海盗”

为日军提供重庆气象信息的间谍案。雅德利（及一位中国女子徐贞）的工作，大大减轻了日军对重庆轰炸造成的伤害。雅德利回到美国后写了本书《中国黑室》（*The Chinese Black Chamber*）介绍这段经历，但是该书直到1983年才被获准解密并出版。从书中的内容可以了解到，当时日本在密码设计上有严重的缺陷。日军和重庆间谍约定的密码本就是美国著名作家赛珍珠（Pearl S. Buck）获得1938年诺贝尔文学奖的《大地》（*The Good Earth*）一书。这本书很容易找到，解密时只要接密码电报的人拿着这本书就能解开密码。密码所在的页数就是一个非常简单的公式：发报日期的月数加上天数，再加上10。比如3月11日发报，密码就在 $3+11+10=24$ 页。这样的密码设计违背了我们前面介绍的“加密函数不应该通过几个自变量和函数值就能推出函数本身”这个原则，这样的密码，破译一篇密文就可能破译以后全部的密文。

该书中还提到日军对保密的技术原理所知甚少。有一次日本的马尼拉使馆向外发报时，发到一半机器卡死，然后居然就照单重发一遍了事，这种同文密电在密码学上是大忌（和我们现在VPN登录用的安全密钥一样，密码机加密时，每次应该自动转一轮，以防同一密钥重复使用，因此即使是同一电文，两次发送的密文也应该是不一样的）。另外，日本外交部在更换新一代密码机时，有些距离远的国家的使馆因为新机器到位较晚，他们居然还使用老机器发送。这样就出现新老机器混用的情况，同样的内容美国会收到新老两套密文，由于日本旧的密码很多已被破解，这样会导致新的密码一出台就毫无机密可言。总的来讲，日本在第二次世界大战中情报经常被美国人破译，他们的海军名将山本五十六（他爸爸56岁时生的他，所以起名五十六）也因此丧命[1]。我们常讲落后是要挨打的，其实不会使用数学也是要挨打的。

1 美国破译了日本的密码，掌握了山本五十六飞机的行踪，然后派战斗机击落了他的座机。

2 信息论时代的密码学

在第二次世界大战中，很多顶尖的科学家包括提出信息论的香农都在为美军情报部门工作，而信息论实际上就是情报学的直接产物。香农提出

信息论后，为密码学的发展带来了新气象。根据信息论，密码的最高境界是敌人在截获密码后，对我方的所知没有任何增加，用信息论的专业术语讲，就是信息量没有增加。一般来讲，当密码之间分布均匀并且统计独立时，提供的信息最少。均匀分布使得敌人无从统计，而统计独立能保证敌人即使看到一段密码和明码后，不能破译另一段密码。这也是《暗算》里传统的破译员老陈破译了一份密报，但无法推广的原因，而数学家黄依依预见到了这个结果，因为她知道敌人新的密码系统编出的密文是统计独立的。

有了信息论后，密码的设计就有了理论基础，现在通用的公开密钥的方法，《暗算》里的“光复一号”密码，应该就是基于这个理论。

公开密钥的原理其实很简单，我们以给上面的单词 Caesar 加解密来说明它的原理。先把它变成一组数，比如它的 ASCII 码 $X = 067097101115097114$（每三位代表一个字母）做明码。现在来设计一个密码系统，对这个明码加密。

1. 找两个很大的素数（质数）P 和 Q，越大越好，比如 100 位长的，然后计算它们的乘积

$$N = P \times Q \qquad (17.1)$$

$$M = (P-1) \times (Q-1) \qquad (17.2)$$

2. 找一个和 M 互素的整数 E，也就是说 M 和 E 除了 1 以外没有公约数。

3. 找一个整数 D，使得 $E \times D$ 除以 M 余 1，即 $E \times D \bmod M = 1$。

现在，世界上先进的、最常用的密码系统就设计好了，其中 E 是公钥，谁都可以用来加密，D 是私钥用于解密，一定要自己保存好。乘积 N 是公开的，即使敌人知道了也没关系。

现在，用下面的公式对 X 加密，得到密码 Y。

$$X^E \ mod \ N = Y \tag{17.3}$$

好了，现在没有密钥 D，神仙也无法从 Y 中恢复 X。如果知道 D，根据费尔马小定理[2]，则只要按下面的公式就可以轻而易举地从 Y 中得到 X。

2 费尔马小定理有两种等价的描述。描述一：P 是一个质数，对于任何整数 N，如果 N、P 互素，那么 $N^{P-1} \equiv 1(mod\ P)$。描述二：P 是一个质数，对于任何整数 $N^P \equiv N(mod\ P)$。

$$Y^D \ mod \ N = X \tag{17.4}$$

这个过程大致可以概况如下：

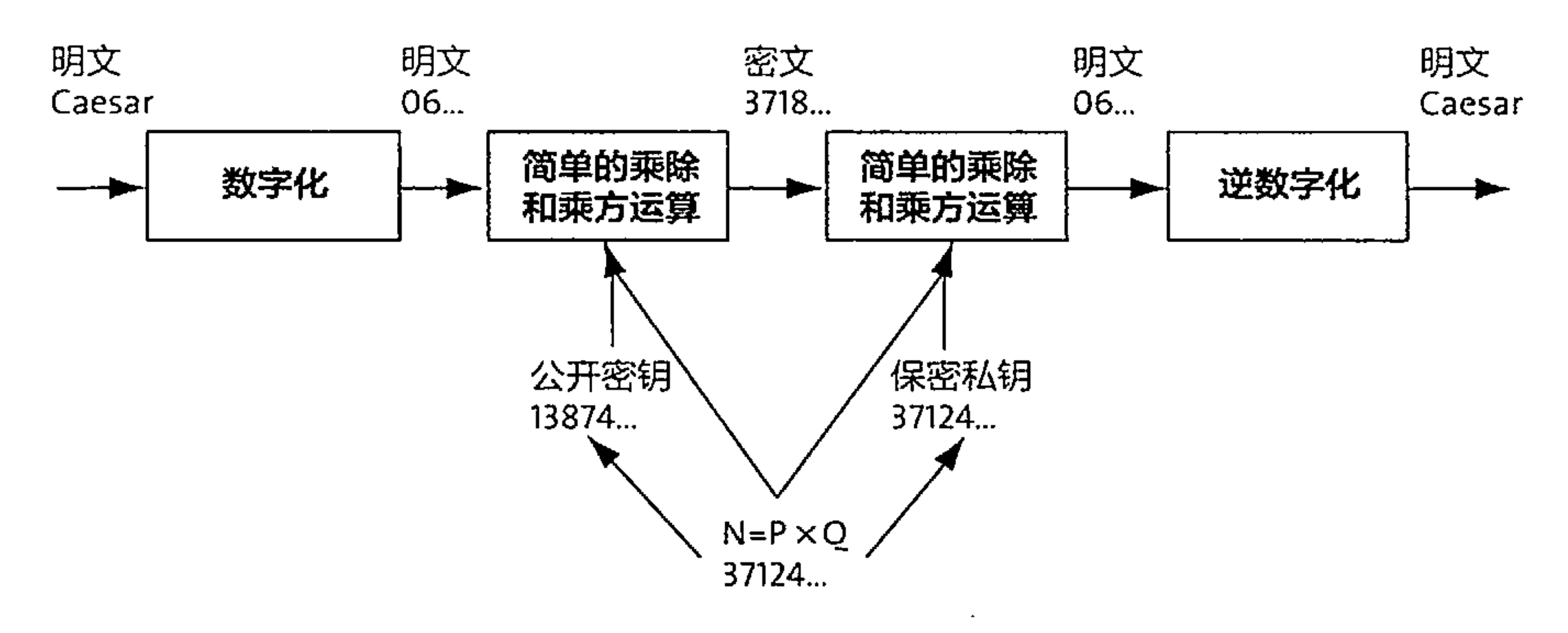


图 17.4　公开密钥示意图

公开密钥的好处有：

1. 简单，就是一些乘除而已。

2. 可靠。公开密钥方法保证产生的密文是统计独立而分布均匀的。也就是说，不论给出多少份明文和对应的密文，也无法根据已知的明文和密文的对应来破译下一份密文。更重要的是 N，E 可以公开给任何人加密用，但是只有掌握密钥 D 的人才可以解密，即使加密者自己也是无法解密的。这样，即使加密者被抓住叛变了，整套密码系统仍然是安全的。（而恺撒大帝的加密方法，只要有一个知道密码本的人泄密，整个密码系统就公开了。）

3. 灵活，可以产生很多的公开密钥 E 和私钥 D 的组合给不同的加密者。

最后让我们看看破解这种密码的难度。首先，要声明，世界上没有永远破不了的密码，关键是它能有多长时间的有效期。要破解公开密钥的加密方式，至今的研究结果表明最好的办法还是对大数 N 进行因数分解，即通过 N 反过来找到 P 和 Q，这样密码就被破解了。而找 P 和 Q 目前只有用计算机把所有的数字试一遍这种笨办法。这实际上是在拼计算机的速度，这也就是为什么 P 和 Q 都需要非常大。一种加密方法只要保证 50 年内计算机破不了也就可以满意了。前几年破解的 RSA-158 密码是这样被因数分解的

395058745832651445264197678006144819960207764603049364541393760515793556265294506836097278424682195350935443058704902519956553357102097992264849779494429556033
= 3388495837466721394368393204672181522815830368604993048084925840555281177 × 1165882340667125990314837655838327081813101225814639260043952099413134433416292453613 9

现在，让我们回到《暗算》中，黄依依第一次找的结果经过一系列计算发现无法归零，也就是说除不尽，我猜她可能试图将一个大数 N 做分解，没成功。第二次计算的结果是归零了，说明她找到 $N = P \times Q$ 的分解方法。当然，这件事恐怕是不能用算盘完成的，所以我觉得编导在这点上有点夸张。另外，该电视剧还有一个讲得不清不楚的地方就是里面提到的“光复一号”密码的误差问题。一个密码是不能有误差的，否则就是有了密钥也无法解码了。我想可能是指在构造密码时，P 和 Q 之一没找对，其中一个（甚至两个都）不小心找成了合数，这时密码的保密性就差了很多。如果谁知道电视剧里面讲的“误差”是指什么，请告诉我一声。再有，电视剧里提到冯·诺依曼，说他是现代密码学的祖宗，这是完全弄错了，应该是香农。冯·诺依曼的贡献在发明现代电子计算机和提出博弈论（Game Theory），和密码无关。

不管怎么样，我们今天用的所谓最可靠的加密方法，背后的数学原理其实就这么简单，一点也不神秘，无非是找几个大素数做一些乘除和乘方运算就可以了。但就是靠这么简单的数学原理，保证了二战后的密码几乎无法被破解。冷战时期美苏双方都投入了前所未有的精力去获得对方的情报，但是没有发生过大的因为密码被破而泄密的事件。

3 小结

我们在介绍信息论中谈到，利用信息可以消除一个系统的不确定性。而利用已经获得的信息情报来消除一个情报系统的不确定性就是解密。因此，密码学的最高境界就是无论敌方获取多少密文，也无法消除己方情报系统的不确定性。为了达到这个目的，就不仅要做到密文之间相互无关，同时密文还是看似完全随机的序列。在信息论诞生后，科学家们沿着这个思路设计出很好的密码系统，而公开密钥是目前最常用的加密办法。

第18章 闪光的不一定是金子——谈谈搜索引擎反作弊问题

自从有了搜索引擎，就有了针对搜索引擎网页排名的作弊（SPAM）。结果用户发现在搜索引擎中排名靠前的网页不一定就是高质量的、相关的网页，而是商业味儿非常浓的作弊网页。用句俗话说，闪光的不一定是金子。

搜索引擎的作弊，虽然方法很多，目的只有一个，就是采用不正当手段提高自己网页的排名。早期最常见的作弊方法是重复关键词。比如一个卖数码相机的网站，重复地罗列各种数码相机的品牌，如尼康、佳能和柯达等等。为了不让读者看到众多讨厌的关键词，聪明一点的作弊者常用很小的字体和与背景相同的颜色来掩盖这些关键词。其实，这种做法很容易被搜索引擎发现并纠正。

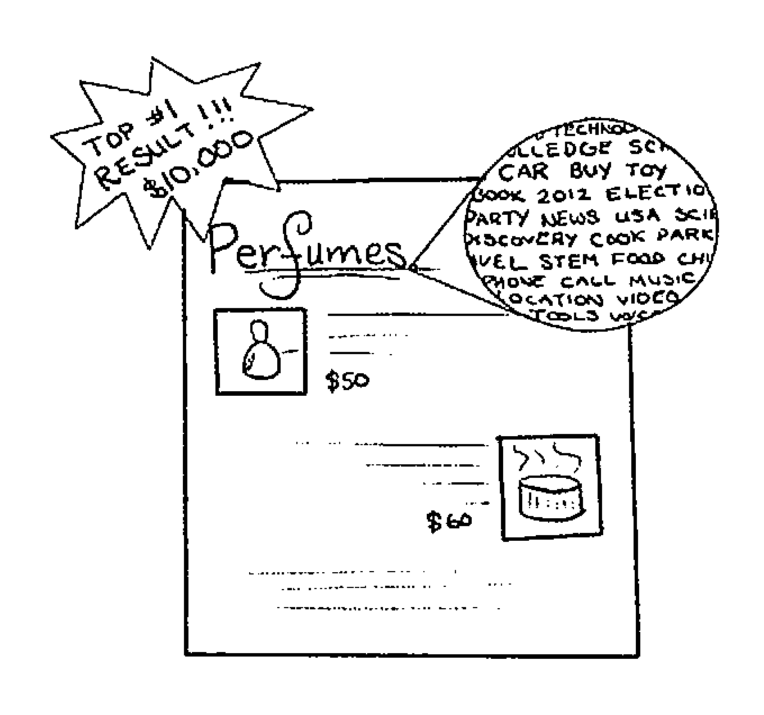


图 18.1 给我 1 万块钱，我保证你的网站在 Google 排在第一页

有了网页排名（PageRank）以后，作弊者发现一个网页被引用的链接越多，排名就可能越靠前，于是就有了专门买卖链接的生意。比如，有人自己创建成百上千个网站，这些网站上没有实质的内容，只有到他们的客户网站的链接。这种做法比重复关键词要高明得多，因为他们自己隐藏在背后，而他们那些客户的网页本身内容上没有什么问题，因此不容易被发现。但是，这些伎俩还是能够被识破的。因为那些所谓帮别人提高排名的网站，为了维持生意需要大量地卖链接，所以很容易露马脚。（这就如同造假钞票，当某一种假钞票的流通量相当大以后，就容易找到源头。）再以后，又有了形形色色的作弊方式，这里就不一一赘述了。

2002年，我加入Google做的第一件事就是消除网络作弊，因为那时搜索引擎的作弊实在太严重。当时全世界没有人做过反作弊的工作，作弊者也不会知道我们要清除他们。经过我们几个人几个月的努力，一举清除了一半的作弊者，并且接下来陆陆续续把绝大多数都清除了。（当然，以后抓作弊的效率就不会有这么高了。)作弊者没有想到我们会清除他们。其中一部分网站从此“痛改前非”，但还是有很多网站换一种作弊方法继续作弊。这些是在我们预料之中的，我们也准备了后招等着他们。因此，抓作弊成了一种长期的“猫捉老鼠”的游戏。虽然至今还没有一个一劳永逸解决作弊问题的方法，但是，Google基本做到了对于任何已知的作弊方法，在一定时间内发现并清除，从而将作弊网站的数量控制在一个很小的比例范围内。

做事情的方法有道和术两种境界，搜索反作弊也是如此。在“术”这个层面的方法大多是看到作弊的例子，分析它，然后清除它，这种方法能解决问题，而且不需要太动脑筋，但是工作量较大，难以从个别现象上升到普遍规律。很多崇尚“人工”的搜索引擎公司喜欢这样的方法。在“道”这个层面解决反作弊问题，就要透过具体的作弊例子，找到作弊的动机和本质。然后，从本质上解决问题。

我们发现，通信模型对于搜索反作弊依然适用。在通信中解决噪音干扰

问题的基本思路有两条。

1. 从信息源出发，加强通信（编码）自身的抗干扰能力。

2. 从传输来看，过滤掉噪音，还原信息。

搜索引擎作弊从本质上看就如同对（搜索）排序的信息加入噪音，因此反作弊的第一条是要增强排序算法的抗噪声能力。其次是像在信号处理中去噪音那样，还原原来真实的排名。学过信息论和有信号处理经验的读者可能知道这么一个事实：如果在发动机很吵的汽车里用手机打电话，对方可能听不清；但是如果知道了汽车发动机的频率，可以加上一个和发动机噪音频率相同，振幅相反的信号，便很容易地消除发动机的噪音，这样，接听人可以完全听不到汽车的噪音。事实上，现在一些高端手机已经有了这种检测和消除噪音的功能。消除噪音的流程可以概括如下：

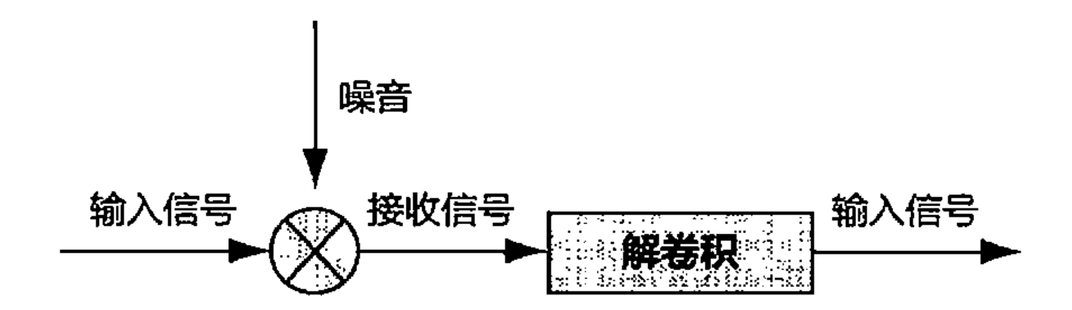


图 18.2　通信中消除噪音的过程

在图 18.2 中，原始的信号混入了噪音，在数学上相当于给两个信号做卷积。噪音消除的过程是一个解卷积的过程。这在信号处理中并不是什么难题。因为第一，汽车发动机的频率是固定的，第二，这个频率的噪音重复出现，只要采集几秒钟的信号进行处理就能做到。从广义上讲，只要噪音不是完全随机并且前后有相关性，就可以检测到并且消除。（事实上，完全随机不相关的高斯白噪音是很难消除的。）

搜索引擎的作弊者所做的事，就如同在手机信号中加入了噪音，使得搜索结果的排名完全乱了。但是，这种人为加入的噪音并不难消除，因为作弊者的方法不可能是随机的（否则就无法提高排名了）。而且，作弊者也不可能是一天换一种方法，即作弊方法是时间相关的。因此，搞搜

索引擎排名算法的人，可以在搜集一段时间的作弊信息后，将作弊者抓出来，还原原有的排名。当然这个过程需要时间，就如同采集汽车发动机的噪音需要时间一样，在这段时间内，作弊者可能会尝到些甜头。因此，有些人看到自己的网站经过所谓的优化（其实是作弊），排名在短期内靠前了，以为这种所谓的优化是有效的。但是，不久就会发现排名掉下去了很多。这倒不是搜索引擎以前宽容，现在严厉了，而是说明抓作弊需要一定的时间，以前只是还没有检测到这些作弊的网站而已。

从动机上讲，作弊者无非是想让自己的网站排名靠前，然后获得商业的利益。而帮助别人作弊的人（他们自称是搜索引擎优化者，Search Engine Optimizer，SEO），他们也是要从中牟利的。掌握了动机就可以针对他们的动机进行防范。具体的做法是，针对和商业相关的搜索，采用一套“抗干扰”强的搜索算法，这就如同在高噪音环境下采用抗干扰的麦克风一样。而对信息类的搜索，采用“敏感”的算法，就如同在安静环境下采用敏感的麦克风，对轻微的声音也能有很好的效果。那些卖链接的网站，都有大量的出链（Out Links），而这些出链的特点和不作弊的网站的出链特点大不相同（可能他们自己不觉得）。每一个网站到其他网站的出链数目可以作为一个向量，它是这个网站固有的特征。既然是向量，我们就可以计算余弦距离。（余弦定理又派上了用处！）我们发现，有些网站的出链向量之间的余弦距离几乎为 1，一般来讲这些网站通常是一个人建的，目的只有一个：卖链接。发现这个规律后，我们改进了 PageRank 算法，使得买来的链接基本上起不到作用。

反作弊用到的另一个工具是图论。在图中，如果有几个节点两两互相都连接在一起，它们称为一个 Clique。作弊的网站一般需要互相链接，以提高自己的排名。这样在互联网这张大图中就形成了一些 Clique。图论中有专门的发现 Clique 的方法，可以直接应用到反作弊中。这里我们再次看到数学的作用。至于术的层面方法则很多，比如针对作弊的 JavaScript 跳转页面[1]，通过解析相应的 JavaScript 内容即可。

1 很多作弊的网页落地页（Landing Page）内容质量非常高，但是里面暗藏一个 JavaScript 跳转到另外一个商业网站。因此，用户进入这个网页后，落地的网页只是一闪而过，就进入到作弊的网页。搜索引擎爬虫下载这个网页后，会按照它高质量的内容建索引。用户查找信息时，这些落地页因为内容较好，就被排在前面，但是用户看到的是和搜索无关的内容。

最后还要强调几点，第一，Google 反作弊和恢复网站原有排名的过程完全是自动的（并没有个人的好恶），就如同手机消除噪音是自动的一样。一个网站要想长期排名靠前，就需要把内容做好，同时要和那些作弊网站划清界限。第二，大部分搜索引擎和帮助别人作弊的人，只针对占市场份额最大的搜索引擎算法来作弊，因为作弊也是有成本的，针对只有市场份额不到 5% 的引擎作弊，在经济上实在不划算。因此，一个小的搜索引擎作弊少，并不一定是它的反作弊技术好，而是到它那里作弊的人太少。

近年来，随着主流搜索引擎对反作弊持续不断的投入，在世界上大多数国家，作弊的成本越来越高了，逐渐赶上甚至超过了花钱在搜索引擎上做广告的费用。现在，希望提高网站排名的商家越来越多地选择通过购买搜索广告的方式来获取流量，而不是作弊。一些体面的网站也和作弊者划清界限。但是在中国恰恰出现了一些相反的趋势：一些网站，包括一些政府网站，为了蝇头小利，出卖链接。这样就诞生了一个灰色收入的行业：收购和贩卖链接的中间商。当然，狐狸穿过草丛，还是要留下痕迹和气味的，这就给了猎人追捕它们的线索。

网页搜索反作弊对于搜索引擎公司来讲是一项长期的任务。作弊的本质是在网页排名信号中加入了噪音，因此反作弊的关键是去噪音。沿着这个思路可以从根本上提高搜索算法抗作弊的能力，事半功倍。而如果只是根据作弊的具体特征头痛医头，脚痛医脚，则很容易被作弊者牵着鼻子走。

第19章 谈谈数学模型的重要性

一直关注“数学之美”系列的读者可能已经发现，我们对任何问题总是在找相应的准确的数学模型。为了说明模型的重要性，2006 年 7 月我在 Google 中国内部讲授搜索基本原理，一共只有 30 学时的课程，却用了整整两个学时来讲数学模型。2010 年我到腾讯后，第一次内部技术讲座也是讲述同样的内容。

在包括哥白尼、伽利略和牛顿在内的所有天文学家中，我最佩服的是地心说的提出者托勒密（Claude Ptolemy，公元 90-168 年）。



图 19.1 伟大的天文学家托勒密

天文学起源于古埃及。由于尼罗河洪水每年泛滥一次，尼罗河下游有着十分肥沃而且灌溉方便的土地，因此孕育出人类最早的农业文明。每当洪水过后，埃及人就在退洪的土地上耕作，然后便可获得很好的收成。这种生产方式一直延续到 20 世纪 60 年代，直到尼罗河上的阿斯旺大坝修成，尼罗河下游再也没有洪水可以灌溉土地为止。（埃及延续了几千年的农业也因此被破坏殆尽。）为了准确预测洪水到来和退去的时间，6 000 年前的埃及人发明了天文学。和我们想象的不同，古埃及人是根据天狼星和太阳在一起的位置来判断一年中的时间和节气。在古埃及的历法中没有闰年，它的一个“季度”也非常长：长达 365 × 4 + 1 = 1 461 天，因为每隔这么多天，太阳和天狼星一起升起。（因此，古埃及的日历是一个周期很长的日历。）事实证明，以天狼星和太阳同时出现做参照系比仅以太阳做参照系更准确些。古埃及人可以准确地判断洪水能到达的边界和时间。

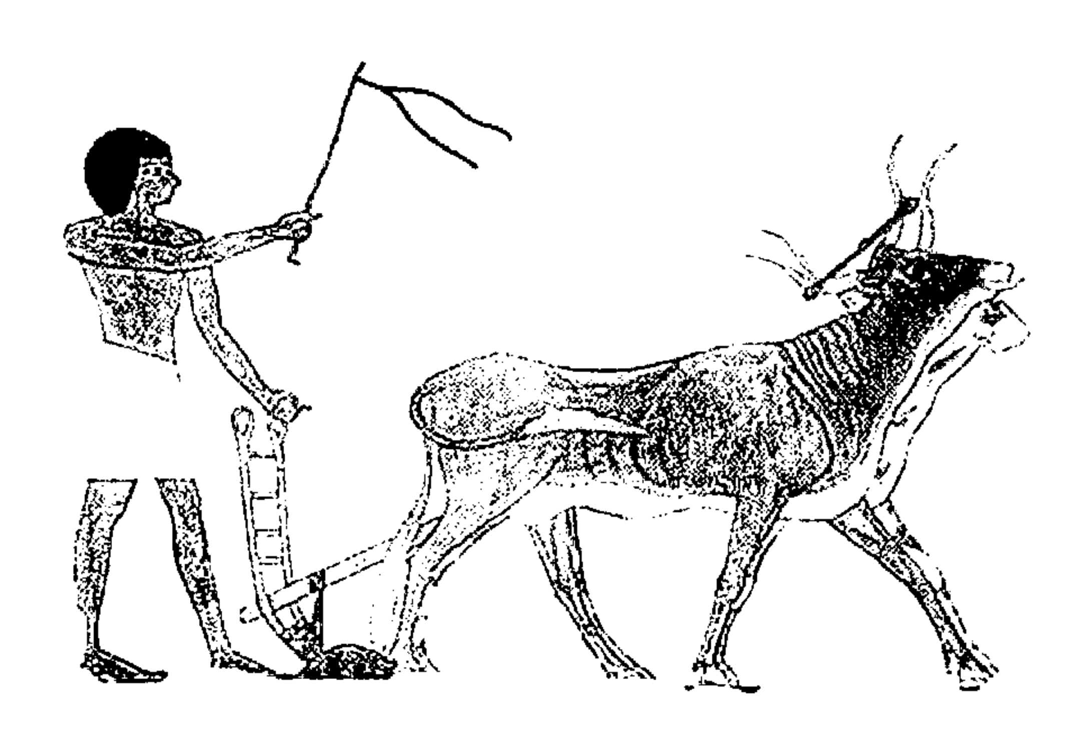


图 19.2 古埃及的农业文明受益于尼罗河的洪水，它直接促进了天文学的诞生

到了人类文明的第二个中心美索不达米亚兴起的时候，那里的古巴比伦人对天文学有了进一步的发展，他们的历法中有了月和四季的概念。同时他们观测到了五大行星（金、木、水、火、土，肉眼看不到天王星和海王星）运行的轨迹不是简单地围绕地球转，而是波浪形地运动。西方语言中行星（Planet）一词的意思就是漂移的星球。他们还观测到行星在近日点运动比远日点快。（图 19.3 是在地球上看到的金星的轨迹，看过《达芬奇密码》一书的读者知道，金星大约每四年在天上画一个五角星。）

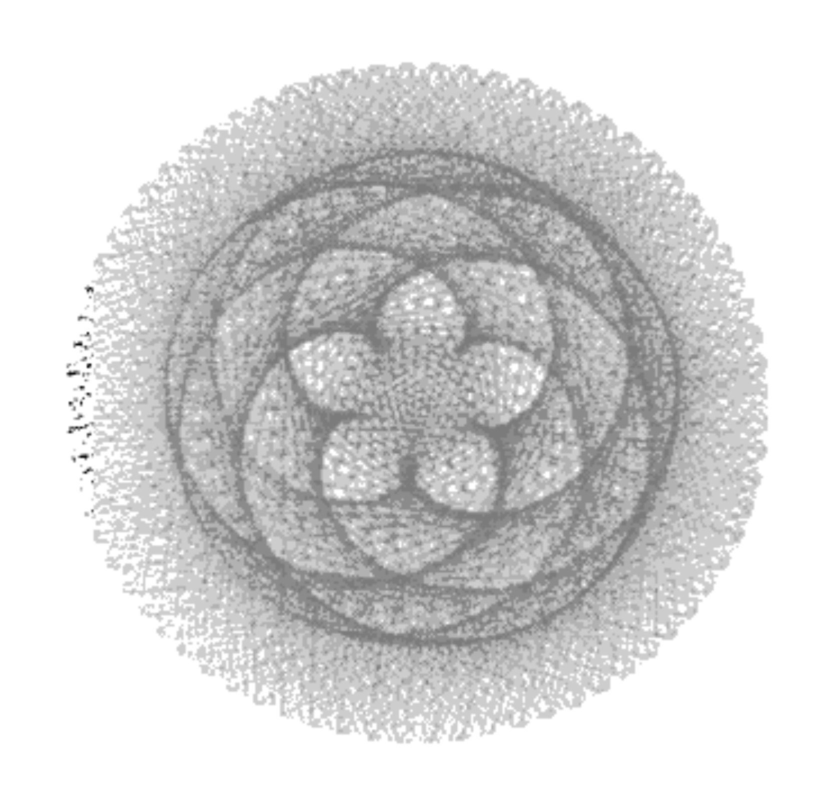


图 19.3　在地球上看到的金星的运动轨迹

但是真正创立了天文学，并且计算出诸多天体运行轨迹的是近两千年前古罗马时代的克劳第斯·托勒密。虽然今天我们可能会嘲笑托勒密犯的简单错误，比如太阳是围绕地球旋转的，但是真正了解托勒密贡献的人都会对他肃然起敬。在过去的几十年里，因为政治的需要，托勒密在中国总是被作为错误理论的代表受到批判，以至于中国人基本上不知道他在人类天文学上无以伦比的贡献。我本人也是在美国读了些科学史的书籍才了解到他的伟大之处。作为数学家和天文学家的托勒密，他有很多发明和贡献，其中任何一项都足以让他在科学史上占有重要的一席之地。托勒密发明了球坐标（我们今天还在用），定义了包括赤道和零度经线在内的经纬线（今天的地图就是这么划的），他提出了黄道，还发明了弧度制（中学生学习的时候可能还会感觉有点抽象）。

当然，他最大也是最有争议的发明是地心说。虽然我们知道地球是围绕太阳运动的，但是在当时，从人们的观测出发，很容易得到地球是宇宙中心的结论。中国古代著名天文学家张衡提出的浑天说，其实就是地心说，但是张衡没有能定量地进行描述。从下面两张图中可以看出两者非常相似。只不过因为张衡是中国人的骄傲，在历史书中从来是正面宣传，而托勒密在中国却成了唯心主义的代表。其实，托勒密在天文学上的地位堪比欧几里德之于几何学，牛顿之于物理学。

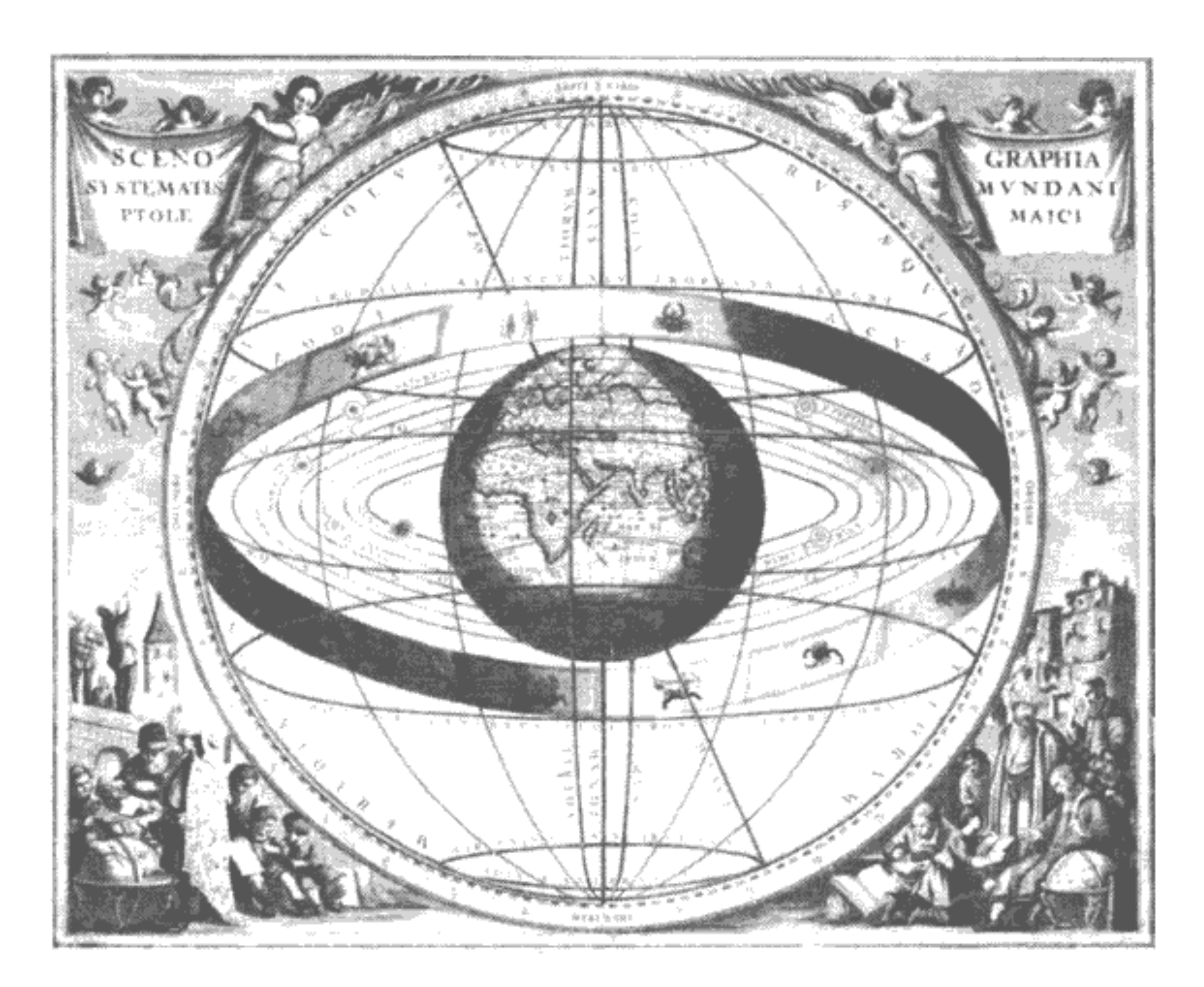


图 19.4　托勒密的地心说模型



图 19.5　张衡的浑天仪（很像地心说的模型）

当然从地球上看，行星的运动轨迹是不规则的，托勒密的伟大之处是用40-60 个小圆套大圆的方法，精确地计算出了所有行星运动的轨迹，如图 19.6 所示。托勒密继承了毕达格拉斯的一些思想，他也认为圆是最完美的几何图形，因此用圆来描述行星运行的规律。

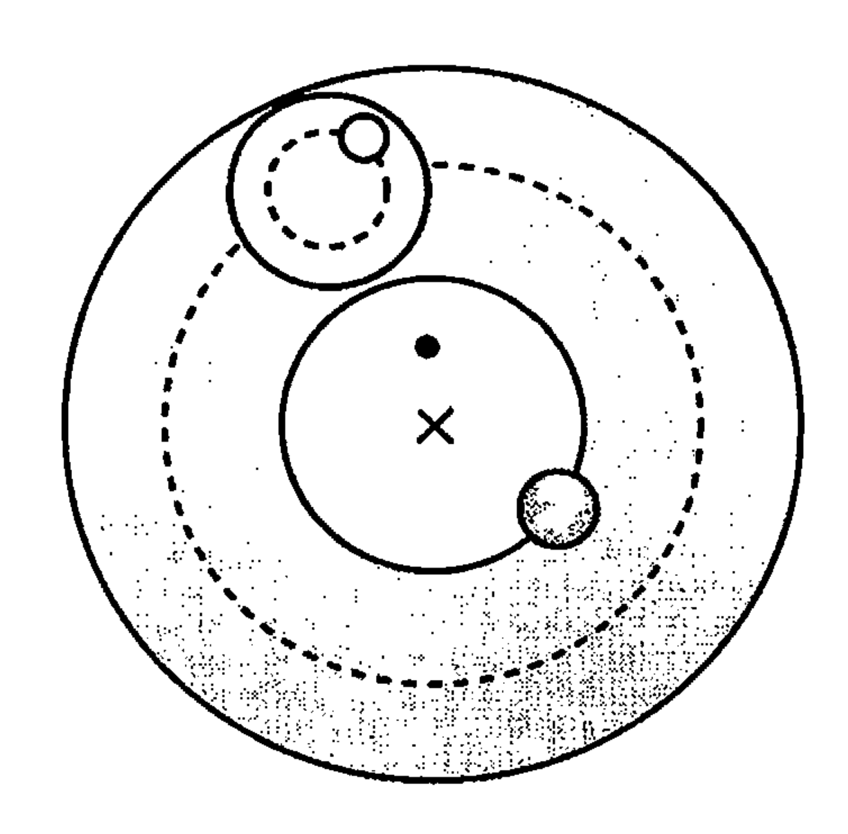


图 19.6 托勒密的小圆套大圆的地心说模型

托勒密模型的精度之高，让后来所有的科学家都惊叹不已。即使今天，我们在计算机的帮助下，也很难解出 40 个套在一起的圆的方程。每每想到这里，我都由衷地佩服托勒密。根据托勒密的计算，制定了儒略历，即每年 365 天，每 4 年增加一个闰年，多一天。1 500 年来，人们根据他的计算决定农时。但是，经过了 1 500 年，托勒密对太阳运动的累积误差，还是多出了 10 天。由于这十天的差别，欧洲的农民从事农业生产差出几乎一个节气，很影响农业。1582 年，教皇格利高里十三世在日历上取消掉 10 天，然后将每一个世纪最后一年的闰年改成平年，然后每 400 年再插回一个闰年，这就是我们今天用的日历，这个日历几乎没有误差。为了纪念格利高里十三世，我们今天的日历也叫做格利高里日历。

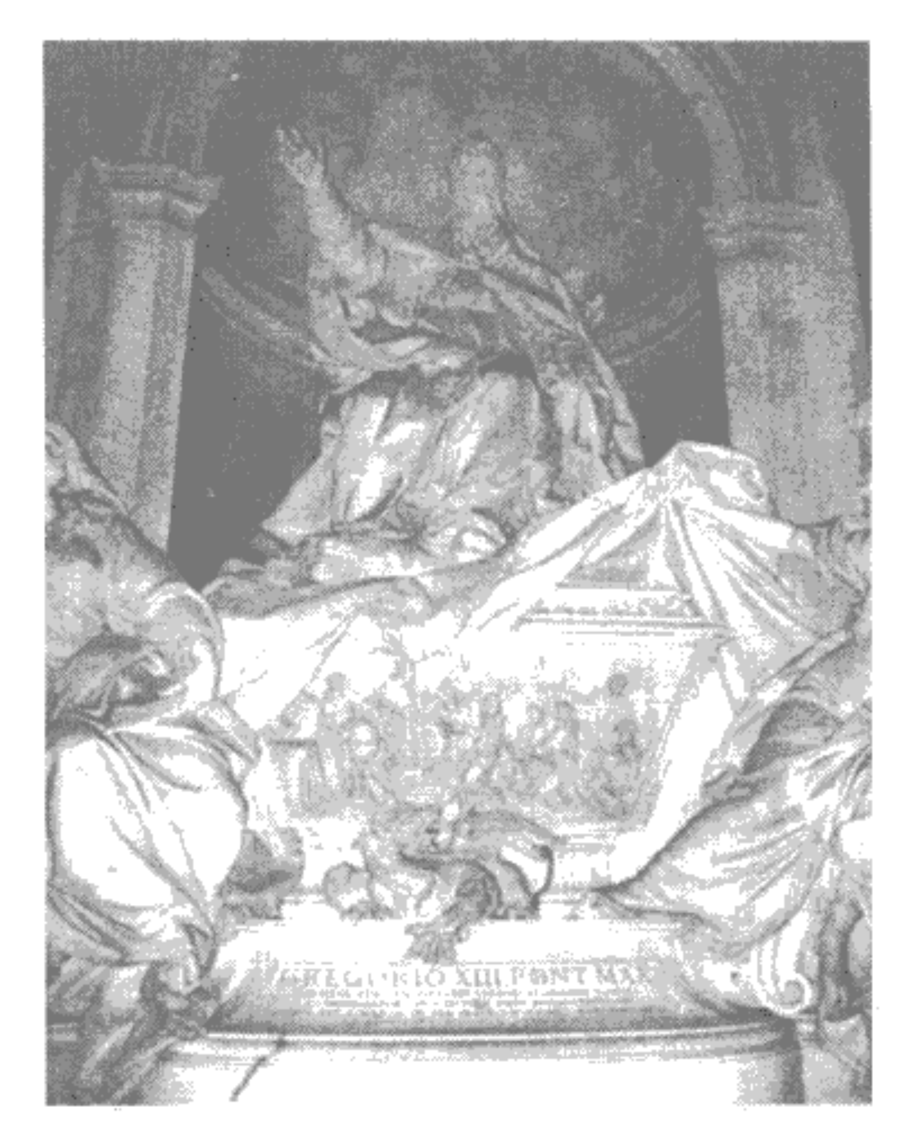


图 19.7 梵蒂冈圣彼得教堂里的格利高里墓，上面的雕像中，教皇格利高里手持新的历法

虽然格利高里十三世“凑出了”准确的历法，即每 400 年比儒略历减去三个闰年。但是教皇并没有从理论上找出原因，因此这种“凑”的做法很难举一反三。格利高里的历法非常准地反映了地球的运动周期，但是对其他行星的运动规律起不到任何帮助。而纠正地心说错误不能是靠在托勒密 40 个圆的模型上再多套上几个圆，而是要进一步探索真理。波兰天文学家哥白尼发现，如果以太阳为中心来描述星体的运行，只需要 8-10 个圆，就能计算出一个行星的运动轨迹，他因此提出了日心说。很遗憾的事，哥白尼正确的假设并没有得到比托勒密更好的结果，他的模型的误差比托勒密模型的误差要大不少。哥白尼生前怕他的日心说惹怒教会，迟迟不敢发表他的学说，直到临终前才发表。而教会初期对这个新的学说的革命性也认识不足，并没有禁止。但是后来当教会发现这个学说有可能挑战上帝创世记的说法时，便开始禁止它了。而哥白尼日心说的不准确性，也是教会和当时的人们认为哥白尼的学说是邪说的另一个重要原因。所以日心说要想让人心服口服地接受，就得更准确地描述行星运动。

完成这一使命的是约翰内斯·开普勒。开普勒在所有一流的天文学家中，资质较差，一生中犯了无数低级的错误。但是他有两样别人没有的东西，首先是从他的老师第谷手中继承的大量的、在当时最精确的观测数据，其次是运气。开普勒很幸运地发现了行星围绕太阳运转的轨道实际上是椭圆形的，这样不需要用多个小圆套大圆，而只要用一个椭圆就能将星体运动规律描述清楚了。开普勒为此提出了三个定律[1]，形式都非常简单，就是三句话。只是开普勒的知识和水平不足以解释为什么行星的轨迹是椭圆形的。

1 开普勒第一定律：行星围绕恒星运动的轨道是一个椭圆，而恒星是这个椭圆的一个焦点。

开普勒第二定律：行星和恒星连线单位时间扫过单位面积。

开普勒第三定律：行星绕太阳公转周期的平方和它们的椭圆轨道的半长轴的立方成正比。

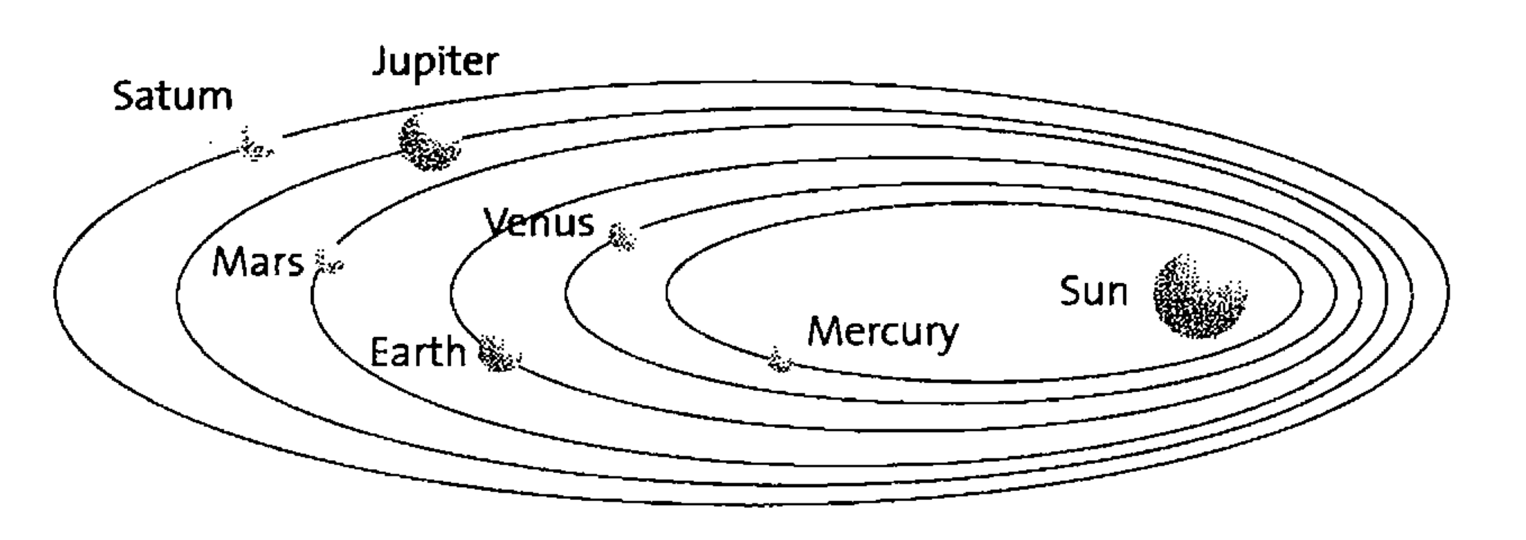


图 19.8　开普勒的行星模型

解释行星运动的轨道为什么是椭圆形这个光荣而艰巨的任务，最后由伟大的科学家牛顿用万有引力定律解释得清清楚楚。

故事到这里似乎可以结束了。但是，许多年后，又有了个小的波澜。1821 年法国天文学家布瓦尔（Alexis Bouvard）发现，天王星的实际轨迹和用椭圆模型算出来的不太符合。当然，偷懒的办法是接着用小圆套大圆的方法修正，但是一些严肃的科学家则努力寻找真正的原因。英国的亚当斯（John Couch Adams）和法国的维内尔（Urbain Le Verrier）在 1861-1862 年间各自独立地发现了吸引天王星偏离轨道的海王星[2]。

2 天文学家伽利略其实早在 1612 年和 1613 年两次观察到海王星，但是他将它误认为是一颗恒星，因此错过了发现海王星的机会。

讲座结束时，我给 Google 中国和腾讯的工程师们总结了下面几个结论：

1. 一个正确的数学模型应当在形式上是简单的。（托勒密的模型显然太复杂。）

2. 一个正确的模型一开始可能还不如一个精雕细琢过的错误模型来的准确，但是，如果我们认定大方向是对的，就应该坚持下去。（日心说开始并没有地心说准确。）

3. 大量准确的数据对研发很重要。（有机会单独介绍。）

4. 正确的模型也可能受噪音干扰，而显得不准确；这时不应该用一种凑合的修正方法来弥补它，而是要找到噪音的根源，这也许能通往重大的发现。

在网络搜索的研发中，我们在前面提到的单文本词频/逆文本频率指数（TF-IDF）和网页排名（PageRank）可以看作是网络搜索中的“椭圆模型”，它们都很简单易懂。

第20章 不要把鸡蛋放到一个篮子里
—— 谈谈最大熵模型

我们在投资时常常讲不要把所有的鸡蛋放在一个篮子里，这样可以降低风险。在信息处理中，这个原理同样适用。在数学上，这个原理称为最大熵原理（The Maximum Entropy Principle）。这是一个非常有意思的题目，但也比较深奥，因此只能大致介绍它的原理。

在网络搜索排名中用到的信息有上百种，腾讯的工程师经常问我如何能把它们结合在一起用好。更普遍地讲，在信息处理中，我们常常知道各种各样但又不完全确定的信息，我们需要用一个统一的模型将这些信息综合起来。如何综合得好，是一门很大的学问。

让我们看一个拼音转汉字的简单例子。假如输入的拼音是“Wang-Xiao-Bo”，利用语言模型，根据有限的上下文（比如前两个词），能给出两个最常见的名字“王小波”和“王晓波”。至于要唯一确定是哪个名字就难了，即使利用较长的上下文也做不到。当然，我们知道，如果通篇文章是介绍文学的，作家王小波的可能性就较大；而在讨论两岸关系时，台湾学者王晓波的可能性会较大。在上面的例子中，只需要综合两类不同的信息，即主题信息和上下文信息。虽然有不少凑合的办法，比如：分为成千上万种不同的主题单独处理，或者对每种信息的作用加权平均等等，但都不能准确而圆满地解决问题，这就好比以前谈到的行星运动

模型中小圆套大圆打补丁的方法。在很多应用中，需要综合几十甚至上百种不同的信息，这种小圆套大圆的方法显然行不通。

1 最大熵原理和最大熵模型

数学上最漂亮的办法是最大熵（Maximum Entropy）模型，它相当于行星运动的椭圆模型。"最大熵"这个名词听起来很深奥，但它的原理很简单，我们每天都在用。说白了，就是要保留全部的不确定性，将风险降到最小。让我们来看一个实际例子。

有一次，我去AT&T实验室作关于最大熵模型的报告，随身带了一个骰子。我问听众"每个面朝上的概率分别是多少"，所有人都说是等概率，即各种点数的概率均为1/6。这种猜测当然是对的。我问听众们为什么，得到的回答是一致的：对这个"一无所知"的骰子，假定它每一个朝上概率均等是最安全的做法。（你不应该主观假设它像韦小宝的骰子一样灌了铅。）从投资的角度看，就是风险最小的做法。从信息论的角度讲，就是保留了最大的不确定性，也就是说让熵达到最大。接着，我又告诉听众，我的这个骰子被我特殊处理过，已知四点朝上的概率是1/3，在这种情况下，每个面朝上的概率是多少？这次，大部分人认为除去四点的概率是1/3，其余的均是2/15，也就是说已知的条件（四点概率为1/3）必须满足，而对其余各点的概率因为仍然无从知道，因此只好认为它们均等。注意，在猜测这两种不同情况下的概率分布时，大家都没有添加任何主观的假设，诸如四点的反面一定是三点等等。（事实上，有的骰子四点的反面不是三点而是一点。）这种基于直觉的猜测之所以准确，是因为它恰好符合了最大熵原理。

最大熵原理指出，需要对一个随机事件的概率分布进行预测时，我们的预测应当满足全部已知的条件，而对未知的情况不要做任何主观假设。（不做主观假设这点很重要）在这种情况下，概率分布最均匀，预测的风险最小。因为这时概率分布的信息熵最大，所以人们称这种模型叫"最大熵模型"。我们常说，不要把所有的鸡蛋放在一个篮子里，其实就是最

大熵原理的一个朴素的说法，因为当我们遇到不确定性时，就要保留各种可能性。

回到刚才谈到的拼音转汉字的例子，我们已知两种信息，第一，根据语言模型，Wang-Xiao-Bo 可以被转换成王晓波和王小波；第二，根据主题，王小波是作家，《黄金时代》的作者等等，而王晓波是台湾研究两岸关系的学者。因此，就可以建立一个最大熵模型，同时满足这两种信息。现在的问题是，这样一个模型是否存在。匈牙利著名数学家、信息论最高奖香农奖得主希萨（I. Csiszar）证明，对任何--组不自相矛盾的信息，这个最大熵模型不仅存在，而且是唯一的。此外，它们都有同一个非常简单的形式 —— 指数函数。下面的公式是根据上下文（前两个词）和主题预测下一个词的最大熵模型，其中 w_3 是要预测的词（王晓波或者王小波），w_1 和 w_2 是它的前两个字（比如说它们分别是“出版”和“小说家”），也就是其上下文的一个大致估计，s 表示主题。

$$P(w_3|w_1,w_2,s)=\frac{1}{Z(w_1,w_2,s)}e^{\lambda_1(w_1,w_2,w_3)+\lambda_2(s,w_3)} \qquad (20.1)$$

其中 Z 是归一化因子，保证概率加起来等于 1。

在上面的公式中，有几个参数 λ 和 Z，它们需要通过观测数据训练出来。我们将在延伸阅读中介绍如何训练最大熵模型的诸多参数。

最大熵模型在形式上是最漂亮、最完美的统计模型，在自然语言处理和金融方面有很多有趣的应用。早期，由于最大熵模型计算量大，试图使用这个模型的科学家一般用一些类似最大熵的近似模型。谁知这一近似，最大熵模型就从完美变得不完美了。结果可想而知，比打补丁的凑合的方法也好不了多少。于是，不少原来热衷于此的学者又放弃了这种方法。第一个在实际信息处理应用中验证了最大熵模型的优势的是宾夕法尼亚大学马库斯的高徒拉纳帕提（Adwait Ratnaparkhi），原 IBM、现任微软的研究员。拉纳帕提的聪明之处在于他没有对最大熵模型进行近似，而是找到了几个最适合用最大熵模型而计算量相对不太大的自然语言处理

问题，比如词性标注和句法分析。拉纳帕提成功地将上下文信息、词性、名词、动词和形容词等句子成分、主谓宾，通过最大熵模型结合起来，做出了当时世界上最好的词性标识系统和句法分析器。拉纳帕提的论文让人们耳目一新。拉纳帕提的词性标注系统，至今仍然是使用单一方法最好的系统。从拉纳帕提的成就中，科学家们又看到了用最大熵模型解决复杂的文字信息处理问题的希望。

在 2000 年前后，由于计算机速度的提升以及训练算法的改进，很多复杂的问题都可以采用最大熵模型了，包括句法分析、语言模型和机器翻译。最大熵模型和一些简单组合特征的模型相比，效果可以提升几个百分点。对于那些对产品质量不是很看重的人和公司来讲，这几个百分点或许不足以给使用者带来明显的感受，但是如果投资的收益能增长哪怕百分之一，获得的利润也是数以亿计的。因此，华尔街向来最喜欢使用新技术来提高他们交易的收益。而证券（股票、债券等）的交易需要考虑非常多的复杂因素，因此，很多对冲基金开始使用最大熵模型，并且取得了很好的效果。

2 最大熵模型的训练

最大熵模型在形式上非常简单，但是在实现上却非常复杂，计算量非常大。假定我们搜索的排序需要考虑 20 种特征，$\{x_1, x_2, \dots, x_{20}\}$，需要排序的网页是 d，那么即使这些特征互相独立，对应的最大熵模型也是“很长”的

$$P(d|x_1, x_2, \dots, x_{20}) = \frac{1}{Z(x_1, x_2, \dots, x_{20})} e^{\lambda_1(x_1,d)+\lambda_2(x_2,d)+\dots+\lambda_{20}(x_{20},d)} \tag{20.2}$$

其中归一化因子

$$Z(x_1, x_2, \dots, x_{20}) = \sum_d e^{\lambda_1(x_1,d)+\lambda_2(x_2,d)+\dots+\lambda_{20}(x_{20},d)} \tag{20.3}$$

这个模型里有很多参数 λ 需要通过模型的训练来获得。

最原始的最大熵模型的训练方法是一种称为通用迭代算法 GIS（Generalized Iterative Scaling）的迭代算法。GIS 的原理并不复杂，大致可以概括为以下几个步骤：

1. 假定第零次迭代的初始模型为等概率的均匀分布。
2. 用第 N 次迭代的模型来估算每种信息特征在训练数据中的分布。如果超过了实际的，就把相应的模型参数变小。否则，将它们变大。
3. 重复步骤 2 直到收敛。

GIS 最早是由达诺奇（J. N. Darroch）和拉特克利夫（D. Ratcliff）在上个世纪 70 年代提出的，它是一个典型的期望值最大化算法（Expectation Maximization，简称 EM）。但是，这两人没能对这种算法的物理含义做出很好的解释。后来是由数学家希萨解释清楚的。因此，人们在谈到这个算法时，总是同时引用达诺奇和拉特克利夫以及希萨的两篇论文。GIS 算法每次迭代的时间都很长，需要迭代很多次才能收敛，而且不太稳定，即使在 64 位计算机上都会出现溢出。因此，在实际应用中很少有人真正使用 GIS。大家只是通过它来了解最大熵模型的算法。

上个世纪 80 年代，天赋异禀的达拉皮垂孪生兄弟（Della Pietra）在 IBM 对 GIS 算法做了两方面改进，提出了改进迭代算法 IIS（Improved Iterative Scaling）。这使得最大熵模型的训练时间缩短了一到两个数量级。这样最大熵模型才有可能变得实用。即使如此，在当时也只有 IBM 有条件使用最大熵模型。

但是，最大熵模型的计算量仍然是个拦路虎。我在学校时花了很长时间考虑如何简化最大熵模型的计算量。有很长一段时间里，我的研究方式就和书呆子陈景润一样，每天一支笔，一沓纸，不停地推导。终于有一天，我对自己的导师说：我发现一种数学变换，可以将大部分最大熵模型的训练时间在 IIS 的基础上减少两个数量级。我在黑板上推导了一个多小时，他没从我的推导中找出任何破绽。接着他又回去想了两天，然后

确认我的算法是对的。从此，我们就构造了一些很大的最大熵模型。这些模型比修修补补的凑合的方法好不少。即使在我找到了快速训练算法以后，为了训练一个包含上下文信息、主题信息和语法信息的文法模型（Language Model），我并行使用了 20 台当时最快的 SUN 工作站，仍然计算了三个月[1]。由此可见最大熵模型复杂的一面。最大熵模型快速算法的实现很复杂。到今天为止，世界上能有效实现这些算法的也不到一百人。有兴趣实现一个最大熵模型的读者可以阅读我的论文[2]。

1 现在用 MapReduce 的工具，在1000台计算机上并行计算，一天就可以完成了。

2 www.cs.jhu.edu/ ~ junwu/publications.html

最大熵模型，可以说是集简繁于一体：形式简单，实现复杂。值得一提的是，在 Google 的很多产品比如机器翻译中，都直接或间接地用到了最大熵模型。

讲到这里，读者也许会问，当年最早改进最大熵模型算法的达拉皮垂兄弟这些年难道没有做任何事吗？他们在上个世纪 90 年代初贾里尼克离开 IBM 后，也退出了学术界，而到金融界大显身手。他们两人和很多 IBM 做语音识别的同事一同到了一家当时还不大，但现在是世界上最成功的对冲基金（Hedge Fund）公司 —— 文艺复兴技术公司（Renaissance Technologies）。我们知道，决定股票涨落的因素可能有几十甚至上百种，而最大熵方法恰恰能找到一个同时满足成千上万种不同条件的模型。在那里，达拉皮垂兄弟等科学家用最大熵模型和其他一些先进的数学工具对股票进行预测，获得了巨大的成功。从该基金 1988 年创立至今，它的净回报率高达平均每年 34%。也就是说，如果 1988 年你在该基金投入一块钱，20 年后的 2008 年你能得到 200 多块钱。这个业绩，远远超过股神巴菲特的旗舰公司伯克希尔哈撒韦（Berkshire Hathaway）。同期，伯克希尔哈撒韦的总回报是 16 倍。而在金融危机的 2008 年，全球股市暴跌，文艺复兴技术公司的回报却高达 80%，可见数学模型的厉害。

3　小结

最大熵模型可以将各种信息整合到一个统一的模型中。它有很多良好的特性：从形式上看，它非常简单，非常优美；从效果上看，它是唯一一种既可以满足各个信息源的限制条件，同时又能保证平滑（Smooth）性的模型。由于最大熵模型具有这些良好的特性，它的应用范围因而十分广泛。但是，最大熵模型的计算量巨大，在工程上实现方法的好坏决定了模型的实用与否。

参考文献：

1. Csiszar, I. I-Divergene Geometry of Probability Distributionsand Minimization Problems. The Annals of Statistics. Vol. 3, No 1, pp.146-158, 1975

2. Csiszar, I. A Geometric Interpretation of Darroch and Ratcliff'sGeneralized Iterative Scaling. The Annals of Statistics. Vol. 17, No.3, pp.1409-1413. 1989

3. Della Pietra, S., Della Pietra, V. &Lafferty, J. Inducing Features of Random Fields, IEEE Trans. on PatternAnalysis and Machine Intelligence. vol.19, No.4, pp280-393, 1997

4. Khudanpur, S. & Wu, J. Maximum Entropy Techniques for Exploiting Syntactic, Semantic and Collocational Dependencies in Language Modeling. Computer Speech and Language Vol.14, No.5, pp.355-372, 2000

5. Wu, J, Maximum entropy language modeling with non-local dependencies, Ph.Ddissertation,
www.cs.jhu.edu/~junwu/publications/dissertation.pdf, 2002

第21章 拼音输入法的数学原理

亚洲语言及所有非罗马拼音式的语言（Non-Roman Languages）的输入原本是个问题，但是近20年来，以中国为代表的亚洲国家在输入法方面有了长足的进步，现在已经不是人们使用计算机的障碍了。以中文输入为例，过去的25年里，输入法基本上经历了以自然音节编码输入，到偏旁笔画拆字输入，再回归自然音节输入的过程。和任何事物的发展一样，这个螺旋式的回归不是简单的重复，而是一种升华。

输入法输入汉字的快慢取决于对汉字编码的平均长度，用通俗的话来讲，就是击键次数乘以寻找这个键所需要的时间。单纯地减少编码长度未必能提高输入速度，因为寻找一个键的时间可能变得较长。提高输入法的效率在于同时优化这两点，而其中有着坚实的数学基础。我们可以通过数学的方法说明平均输入一个汉字需要多少次击键，如何编码能够使得输入法接近理论上的最小值，同时寻找一个键的时间又不至于过长。

1 输入法与编码

将一个方块形状的汉字输入到计算机中，本质上是一个将我们人为约定的信息记录编码 —— 汉字，转换成计算机约定的编码（国标码或者UTF-8码）的信息转换过程。键盘是一种主要的输入工具，当然还可以

有其他输入工具，比如手写板和麦克风。一般来讲，键盘上可使用的只有26个字母加上10个数字键作为对汉字编码的基本键，外加一些控制键。因此，最直接的编码方式就是让这26个字母对应拼音，当然，为了解决汉字的一音多字问题，得用10个数字键来消除歧义性。

这里面，对汉字的编码分为两部分：对拼音的编码（参照汉语拼音标准即可）和消除歧义性的编码。对一个汉字编码的长度取决于这两方面，只有当这两个编码都缩短时，汉字的输入才能够变快。早期的输入法常常只注重第一部分而忽视第二部分。

虽然全拼输入法和汉语拼音标准一致，容易学习，但是，拼音输入法早期甚至是双拼早于全拼，原因是为了缩短对拼音的编码。在双拼输入法中，每个声母和韵母只用一个键即可表示。中国最早可以输入汉字的微机中华学习机和长城0520，分别对应苹果系列和IBM系列，采用的都是双拼的输入方案。台湾地区用的注音字母也等效于双拼。各家的双拼对应键盘字母的方式还略有不同，以微软公司为例，对应如下：

表21.1 声母和键盘字母对应表

韵母	iu	ua	er,uan,van	ue	uai	uo	un,vn	ong,iong
键盘字母	q	w	r	t	y	o	p	s
韵母	uang,iang	en	eng	ang	an	ao	ai	ing
键盘字母	d	f	g	h	j	k	l	;
韵母	ei	ie	iao	ui,ue	ou	in	iam	
键盘字母	z	x	c	v	b	n	m	

这些输入方法看似节省了一点编码长度，但是输入一点也不快，因为它们只优化了局部，而伤害了整体。首先，双拼输入法增加了编码上的歧义性：键盘的字母只有26个，可是汉语的声母和韵母总和却有50多个。从上表中可以看到，很多韵母不得不共享一个字母键。增加歧义性的结果就是从更多汉字候选中找到自己想输入的字，也就是增加消除歧义性编码的长度：不断地重复“翻页，扫描后续字”的过程。第二，它

增加了每一次击键的时间。因为双拼的方法不自然，比全拼的方法多出来一道将读音拆成声母和韵母编码的过程。认知科学的研究表明，在脱稿输入时，拆字的过程会使得思维变慢。第三，双拼对读音的容错性不好，因为前鼻音 an、en、in 和对应的后鼻音 ang、eng、ing，卷舌音 ch、sh、zh 和相应的平舌音（非卷舌音）编码完全没有相似性。全中国除了北京周围的人，大部分人前鼻音和后鼻音、卷舌音和非卷舌音多少有点分不清，经常出现输入韵母和声母后，翻了好几页，也找不到自己想要的字的情况。原因是一开始就选错了韵母或者声母。一个好的输入法不能要求用户一定得把每个字的音都读准，就如同一架普及型的照相机不应该要求使用者精通光圈和快门速度的设置。

由于种种原因，早期的拼音输入法不是很成功，这就给其它输入法的迅速崛起创造了条件。很快，各种输入法如雨后春笋般地冒了出来，总数上，有的报道说有上千种，有的报道说有三千多种。各种输入法的专利到 20 世纪 90 年初已经有上千件，以至于一些专家认为中国软件行业之所以上不去，是因为大家都去做输入法了。所有这些输入法，除了少数对拼音输入法的改进，大多是利用 26 个字母和 10 个数字对汉字库（当时一般只考虑二级国标汉字）中 6 300 个左右的常见字直接编码。大家知道，即使只用 26 个字母编码，三个键的组合也可以表示 $26^3 \approx 17000$ 个汉字，因此，所有这些编码方法都宣称自己能两三个键就输入一个汉字，常见字两个键，非常见字三个键也足够了。其实这里面没有什么学问，很容易做到。但是，这些复杂的编码要让人记住几乎是不可能的，因此这里面的艺术就是如何将编码和汉字的偏旁、笔画或者读音结合，让人记住。当然，每一种编码都宣称自己比其他方法更合理，输入更快。因为这些输入法的编码方法从信息论的角度来看都在同一个水平，互相也比不出什么优劣。但是为了证明自己的方法比别人的快，大家继续走偏，单纯追求击键次数少，最直接的方法就是对词组进行编码，但这样一来，使用者就更无法记住了，只有这些输入法的表演者能记住。这已经不是技术的比赛，而是市场的竞争。最后，王永民的五笔输入法暂时胜出，但

并不是他的编码方法更合理，而是他比其他发明者（大多数是书呆子）更会做市场而已。现在，即使五笔输入法也已经没有多少市场了，这一批发明人可以说是全军覆没。

这一代输入法的问题在于减少了每个汉字击键的次数，而忽视了找到每个键的时间。要求非专业使用者背下这些输入方法里所有汉字的编码是不现实的，这比背 6 000 个 GRE 单词还难。因此，他们在使用这些输入方法时都要按照规则临时“拆字”，即找到一个字的编码组合，这个时间不仅长，而且在脱稿打字时严重中断思维。本书一开头就强调语言和文字作为通信的编码手段，一个重要目的是帮助思维和记忆。如果一个输入法中断了人们的思维过程，就和人的自然行为不相符合。认知科学已经证明，人一心无二用。过去在研究语音识别时做过很多用户测试，发现使用各种复杂编码输入法的人在脱稿打字时，速度只有他在看稿打字时的一半到四分之一。因此，虽然每个字平均敲键次数少，但是敲键盘的速度也慢了很多，总的来看并不快。因此，广大中国计算机用户对这一类输入法的认可度极低，这是自然选择的结果。

最终，用户还是选择了拼音输入法，而且是每个单词编码较长的全拼输入法。虽然看上去这种方法输入每个汉字需要多敲几个字，但是有三个优点让它的输入速度并不慢。第一，它不需要专门学习。第二，输入自然，不会中断思维，也就是说找每个键的时间非常短。第三，因为编码长，有信息冗余量，容错性好。比如对分不清非前鼻音 an、en、in 和后鼻音 ang、eng、ing 的人来讲，输入 zhan（占）这个字，即使他以为拼音是卷舌音 zhang，当输入一半的时候，已经看到自己要找的字了，就会停下来，避免了双拼的那种不容错的问题。当然，拼音输入法要解决的问题只剩下排除一音多字的歧义性了，只要这个问题解决了，拼音输入法照样能做到击键次数和那些拆字的方法差不多，这也是目前各种拼音输入法做的主要工作。接下来我们就分析平均输入一个汉字可以做到最少几次击键。

2 输入一个汉字需要敲多少个键 —— 谈谈香农第一定理

从理论上分析，输入汉字到底能有多快？这里需要用到信息论中的香农第一定律。

假定在国标 GB2312 里面，一共有 6 700 多个常用的汉字。如果不考虑汉字频率的分布，用键盘上的 26 个字母对汉字进行编码，两个字母的组合只能对 676 个汉字编码，对 6 700 多个汉字进行编码需要用三个字母的组合，即编码长度为三。当然，聪明的读者马上发现了我们可以对更常见的字用较短的编码，对不太常见的字用较长的编码，这样平均下来，每个汉字的编码长度可以缩短。假定每一个汉字出现的相对频率是

$$p_1, p_2, p_3, \cdots, p_{6700} \qquad (21.1)$$

它们编码的长度是

$$L_1, L_2, L_3, \cdots, L_{6700} \qquad (21.2)$$

那么，平均编码长度是

$$p_1 \cdot L_1 + p_2 \cdot L_2 + p_3 \cdot L_3 + \cdots + p_{6700} \cdot L_{6700} \qquad (21.3)$$

香农第一定理指出：对于一个信息，任何编码的长度都不小于它的信息熵。因此，上面平均编码长度的最小值就是汉字的信息熵，任何输入法不可能突破信息熵给定的极限。这里需要指出的是，如果将输入法的字库从国标 GB2312 扩展到更大的字库 GBK，由于后者非常见字的频率非常小，平均编码长度比针对国标的大不了多少。因此本书中就以国标字库为准。

现在让我们来回忆一下汉字的信息熵（见第 6 章“信息的度量和作用”）

$$H = -p_1 \cdot \log p_1 - p_2 \cdot \log p_2 - \cdots - p_{6700} \cdot \log p_{6700} \qquad (21.4)$$

如果对每一个字进行统计，而且不考虑上下文相关性，大致可以估算出

它的值在 10 比特以内，当然这取决于用什么语料库来做估计。如果假定输入法只能用 26 个字母输入，那么每个字母可以代表 log26 ≈ 4.7 比特的信息，也就是说，输入一个汉字平均需要敲 10 / 4.7 ≈ 2.1 次键。

聪明的读者也许已经发现，如果把汉字组成词，再以词为单位统计信息熵，那么，每个汉字的平均信息熵将会减少。这样，平均输入一个字可以少敲零点几次键盘。不考虑词的上下文相关性，以词为单位统计，汉字的信息熵大约是 8 比特左右，也就是说，以词为单位输入一个汉字平均只需要敲 8 / 4.7 ≈ 1.7 次键。这就是现在所有输入法都是基于词输入的根本原因。当然，如果再考虑上下文的相关性，对汉语建立一个基于词的统计语言模型（见第 3 章“统计语言模型”），可以将每个汉字的信息熵降到 6 比特左右，这时，输入一个汉字只要敲 6 / 4.7 ≈ 1.3 次键。如果一种输入方法能做到这一点，那么汉字的输入已经比英文快得多了。

但是，事实上没有一种输入方法接近这个效率。这里面有两个主要原因。首先，要接近信息论给的这个极限，就要对汉字的词组根据其词频进行特殊编码。我们在上一节中讲到，过于特殊的编码其实欲速不达。其次，在个人计算机上，很难安装非常大的语言模型。因此，这种编码方法理论上讲有效，但不实用。

现在看看全拼的拼音输入法能够做到输入一个汉字平均多少次击键。汉语全拼的平均长度为 2.98，只要基于拼音的输入法能利用上下文彻底解决一音多字的问题，平均每个汉字输入的敲键次数应该在三次以内，每分钟输入 100 个字完全有可能达到。如果能够更多地利用上下文相关性，可以做到当句子中一个汉字的拼音敲完一部分的时候，这个汉字就被提示出来，因此，全拼输入法的平均击键次数应该小于 3。

因此，接下来的任务就是如何利用上下文了。10 年前的拼音输入法（以紫光为代表）解决这个问题的办法是建立大词库，词也越来越多，越来越长，最后把整句唐诗都作为一个词。这个办法多少能解决一些问题，但如果统计一下就会发现帮助并不大。因为在汉语中，虽然长词的数量

可以非常多，多到几十万，但是一字词和二字词占文本的大多数。而一字词和二字词恰恰是一音多字情况最严重的，比如“zhi”有 275 个（以 Google 拼音输入法统计），二字词“shi-yan”有 14 个。这些不是简单增加词典大小能解决的。增大词典，多少是根据经验和直觉自发的行为，这就和地心说在无法解释行星不规律运动轨迹时，用大圆套了几个小圆的方法凑出了结果一样。为了解决这些问题，接下来很多输入法，包括目前一些很流行的输入法，把常见的词组和词的组合（比如“我是”）放到词典中去。但是汉语有四五万常用的一字词和二字词，这些词的合理组合是上千万乃至上亿，总不能都放到词典中吧。因此枚举词的组合的做法，如同在小圆上再套一些更小的圆，其结果可能距离真理更近，但依然不是真理。

而利用上下文最好的办法是借助语言模型。只要承认概率论，就无法否认语言模型可以保证拼音转汉字（解决一音多字问题）的效果最好。假定有大小不受限制的语言模型，是可以达到信息论给出的极限输入速度的。但是在产品中，不可能占用用户太多的内存空间，因此各种输入法只能提供给用户一个压缩得很厉害的语音模型。而有的输入法为了减小内存占用，或者没有完全掌握拼音转汉字的解码技巧，根本就没有语言模型。这样一来，当今的输入法和极限输入速度相比还有不少可提升的空间。目前，各家拼音输入法（Google 的、腾讯的和搜狗的）基本处在同一个量级，将来技术上进一步提升的关键就在于看谁能准确而有效地建立语言模型。当然利用语言模型将拼音串自动转成汉字，要有合适的算法，这就是下一节要讨论的问题。

3 拼音转汉字的算法

拼音转汉字的算法和在导航中寻找最短路径的算法相同，都是动态规划。这听起来多少有点牵强，拼音输入法和导航又有什么关系呢？

其实可以将汉语输入看成一个通信问题，而输入法则是一个将拼音串变

到汉字串的转换器。每一个拼音可以对应多个汉字，把一个拼音串对应的汉字从左到右连起来，就是一张有向图，它被称为网格图或者篱笆图（Lattice），形式如下：

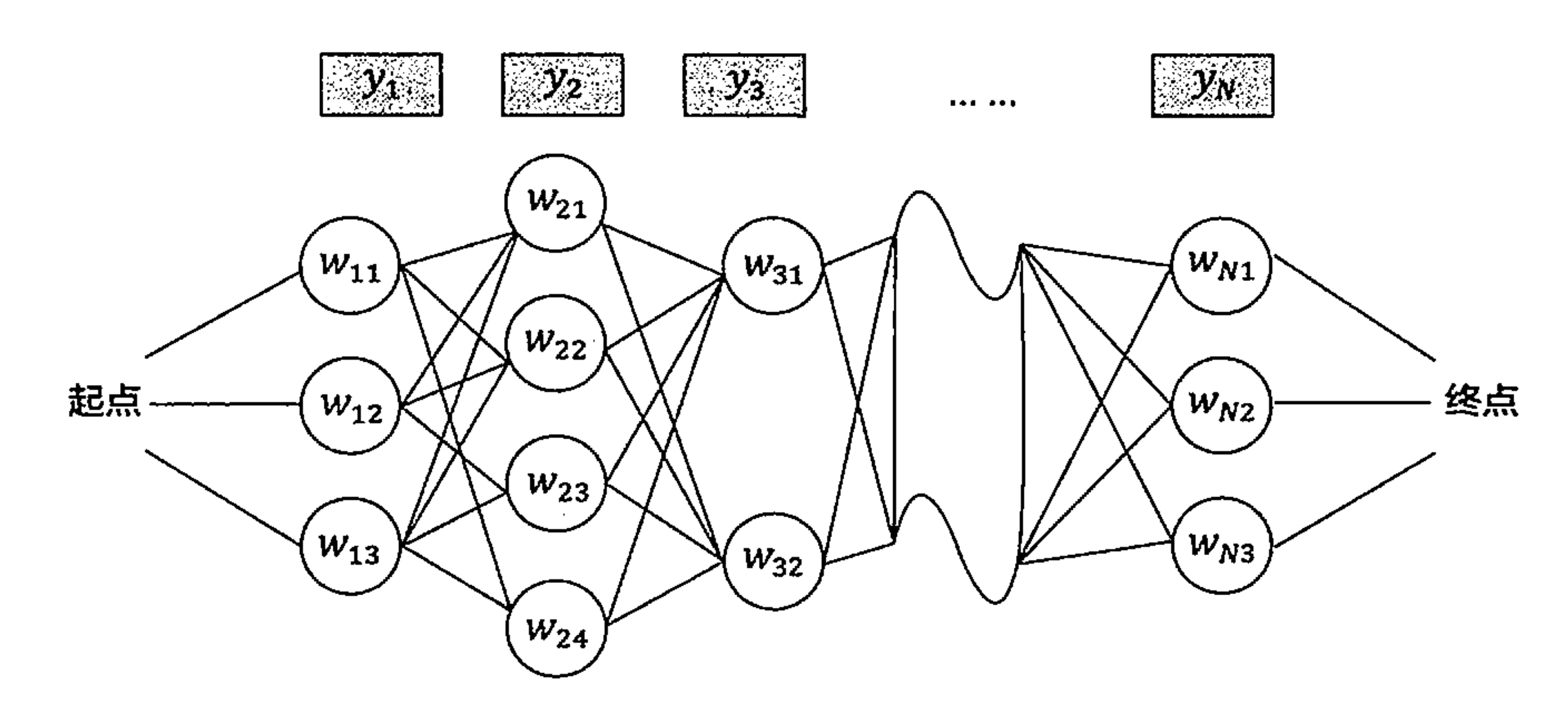


图 21.1　拼音到汉字转换解码的网格图

其中，$y_1, y_2, y_3, \cdots, y_N$是使用者输入的拼音串；$w_{11}, w_{12}, w_{13}$是第一个音 y_1 的候选汉字（我们在后面的公式中用变量w_1代表这三个候选汉字）；$w_{21}, w_{22}, w_{23}, w_{24}$是对应于$y_2$的候选汉字，以变量$w_2$统一代表，以此类推。从第一个字到最后一个字可以组成很多很多句子，每一个句子和图中的一条路径一一对应。拼音输入法就是要根据上下文在给定拼音条件下找到一个最优的句子，即

$$w_1, w_2, \cdots, w_N = \underset{w \in W}{\text{ArgMax}}\, P(w_1, w_2, \cdots, w_N | y_1, y_2, \cdots, y_N) \qquad (21.5)$$

对应到上图中，就是要找从起点到终点的一条最短路径。而要寻找最短路径，首先要定义图中两个节点之间的距离。回顾我们在“隐含马尔可夫模型”一章中介绍上面公式的简化方法，

$$\begin{aligned} & w_1, w_2, \cdots, w_N \\ & \quad = \underset{w \in W}{\text{ArgMax}}\, P(y_1, y_2, \cdots, y_N | w_1, w_2, \cdots, w_N) \cdot P(w_1, w_2, \cdots, w_N) \\ & \quad \approx \underset{w \in W}{\text{ArgMax}} \prod_{i=1}^{N} P(w_i | w_{i-1}) \cdot P(y_i | w_i) \end{aligned} \qquad (21.6)$$

如果对上面公式中的概率取对数同时取反，即定义 $d(w_{i-1}, w_i) = -\log P(w_i|w_{i-1}) \cdot P(y_i|w_i)$，上面的连乘关系变成加法，寻找最大概率的问题就变成了寻找最短路径的问题。这样就可以直接利用动态规划算法实现拼音输入法中最重要的拼音到汉字的转换问题。对比在卫星导航中寻找两个地点之间的最短距离，我们发现它们可以完全对应起来。唯一的差别就是在导航图中，节点（城市）之间是真实的距离，而在拼音串转为汉字的网格图中，两个节点（词）w_{i-1} 和 w_i 之间的距离是转移概率和生成概率的乘积 $-\log P(w_i|w_{i-1}) \cdot P(y_i|w_i)$。

这个拼音输入法的例子和前面第 12 章“地图和本地搜索的最基本技术 —— 有限状态机和动态规划”中提到的导航系统看似没什么关系，但是其背后的数学模型却完全一样。数学的妙处在于它的每一个工具都具有相当的普遍性，在不同的应用中都可以发挥很大的作用。

4　延伸阅读：个性化的语言模型

读者知识背景：概率论。

现有的汉字拼音输入法距离信息论给的极限还有很大的差距，可提升的空间很大，会有越来越好用的输入方法不断涌现。当然，速度只是输入法的一个而不是唯一的衡量标准 —— 当输入速度超过一定阈值后，用户的体验可能更重要。

从理论上讲，只要语言模型足够大，拼音输入法的平均击键次数就可以接近信息论给的极限值。如果把输入法放在云计算上，这是完全可以实现的，而在客户端上（比如个人电脑上）这样做不现实。好在客户端有客户端的优势，可以建立个性化的语言模型。

个性化的出发点是不同人平时写的东西主题不同，由于文化程度的差异，用的词也不同，说话和写作的水平也不同，因此，他们各自应该有各自

的语言模型。

我们在 Google 统计发现，这个假设是对的，且不说表达主题意义的实词会因人而异，就是不同的地方、不同的文化背景和受教育程度的人使用的虚词都有明显的不同。因此，如果每个人有各自的语言模型，用拼音输入时，候选词次序的排列一定比通用的输入法排得好。

接下来有两个问题需要解决。首先是如何训练一个个性化的语言模型，其次是处理好它和通用语言模型的关系。

为每个人训练一个特定的语言模型，最好是收集足够多这个人自己写的文字，但是一个人一辈子写的东西不足以训练一个语言模型。训练一个词汇量在几万的二元模型，需要几千万词的语料，即使是职业作家或者记者，一辈子都不可能写这么多的文章。没有足够多的训练数据，训练出的（高阶）语言模型基本上没有用。当然训练一个一元模型不需要太多数据，一些输入法（自发地）找到了一种经验做法：用户词典，这实际上是一个小规模的一元模型加上非常小量的N元组（比如一个用户定义的词 ABC，实际是一个三元组）。

更好的办法是找到大量符合用户经常输入的内容和用语习惯的语料，训练一个用户特定的语言模型。这里面的关键显然是如何找到这些符合条件的语料。这次又要用到余弦定理和文本分类的技术了。整个训练用户特定的语言模型步骤如下：

1. 可以将训练语言模型的文本按照主题分成很多不同的类别，比如 1 000 个，$C_1, C_2, \dots, C_{1000}$。

2. 对于每个类，找到它们的特征向量（TF-IDF）$X_1, X_2, X_3, \dots, X_{1000}$。这两条我们在以前都讲过。

3. 统计某个人输入的文本，得到他输入的词的特征向量Y。

4. 计算Y和$X_1, X_2, \dots, X_{1000}$的余弦（距离）。

5. 选择前K个和Y距离最近的类对应的文本，作为这个特定用户语言模型的训练数据。

6. 训练一个用户特定的语言模型M_1。

在大部分情况下，M_1对这个特定用户的输入比通用模型M_0好。但是对于相对偏僻的内容，M_1的效果就远不如通用的模型M_0了，因为M_1的训练数据比M_0的小一两个数量级，覆盖的语言现象少得多。因此，一个更好的办法就是综合这两个模型。

我们在最大熵模型中介绍过，把各种特征综合在一起最好的方法是采用最大熵模型。当然，这个模型比较复杂，训练时间较长，如果为每一个人都建立这样一个模型，成本较高。因此可以采用一个简化的模型：线性插值的模型。

假定M_0和M_1都是二元模型，它们计算出的(w_{i-1}, w_i)的条件概率分别是$P_0(w_i|w_{i-1})$和$P_1(w_i|w_{i-1})$。新的模型为M'，计算的条件概率应该是

$$P'(w_i|w_{i-1}) = \lambda(w_{i-1}) \cdot P_0(w_i|w_{i-1}) + (1 - \lambda(w_{i-1})) \cdot P_1(w_i|w_{i-1})$$

其中，$0 < \lambda(w_{i-1}) < 1$是一个插值参数。由于信息熵（对应语言模型的复杂度）是一个凸函数[1]，线性组合P'的熵比P_0和P_1熵的线性组合小，因此新的组合模型不确定性少，是更好的模型。也就是说，将个性化模型和原来的通用模型组合，得到的新模型更好。

这种线性插值的模型比最大熵模型效果略差，但是能得到大约 80% 的收益。（如果最大熵模型比原来的通用模型的改进收益是 100% 的话。）顺便说一句，Google 拼音的个性化语言模型就是这么实现的。

1 凸函数的定义：如果一个函数f，满足条件$f(tx_1 + (1-t)x_2) < tf(x_1) + (1-t)f(x_2)$，那么这个函数称为凸函数。

5 小结

汉字的输入过程本身就是人和计算机的通信，好的输入法会自觉或者不自觉地遵循通信的数学模型。当然要做出最有效的输入法，应当自觉使用信息论做指导。

第22章　自然语言处理的教父马库斯和他的优秀弟子们

1　教父马库斯

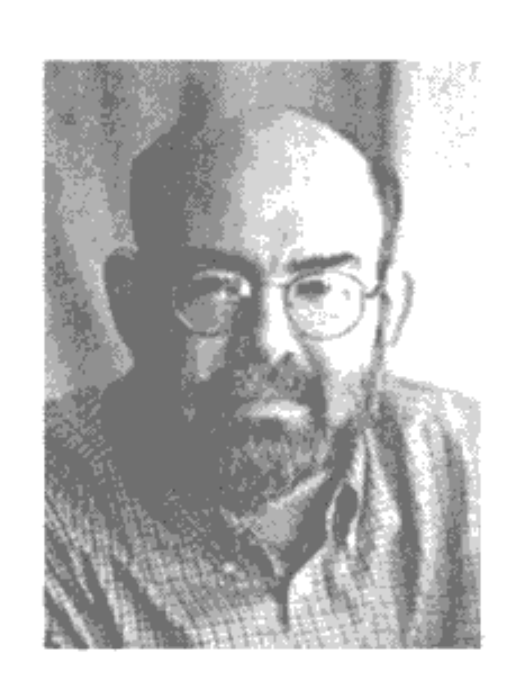


图 22.1　马库斯

将自然语言处理从基于规则的研究方法转到基于统计的研究方法上，贡献最大的有两个人，一个是我们前面介绍过的贾里尼克，他是一位开创性人物，另一个是将这个研究方法进一步发扬光大的米奇·马库斯（Mitch Marcus）。和贾里尼克不同，马库斯对这个领域的贡献不是直接的发明，而是通过他造福于全世界研究者的宾夕法尼亚大学 LDC 语料库以及他的众多优秀弟子。这些优秀弟子，包括我前面章节中介绍的一大批年轻有为的科学家，迈克尔·柯林斯（Michael Collins），艾里克·布莱尔（Eric Brill），大卫·雅让斯基（David Yarowsky），拉纳帕提（Adwait

Ratnaparkhi），以及很多在麻省理工学院和约翰·霍普金斯大学等世界一流大学和 IBM 等公司的研究所担任终身教职和研究员的科学家。就像许多武侠小说中描写的，弟子都成了各派的掌门，师傅一定了不得。的确，马库斯虽然作为第一作者发表的论文并不多，但是从很多角度上讲，他可以说是自然语言处理领域的教父。

和贾里尼克一样，马库斯也毕业于麻省理工学院，同样，他也经历了从工业界（AT&T 贝尔实验室）到学术界（宾夕法尼亚大学）的转移。刚到宾夕法尼亚大学时，马库斯在利用统计的方法进行句子分析上做出了不少成绩。而这方面恰恰是贾里尼克和 IBM 的科学家没有做过的。在马库斯以前，基于统计的自然语言处理为语言学术界所诟病的一个原因是统计的方法很难进行"深入的"分析，马库斯的工作证明统计的方法比规则的方法更适合对自然语言做深入的分析。但是，随着工作的深入以及研究结果的不断进步，马库斯发现两大问题：首先，可以用于研究的统计数据明显不够；其次，各国科学家因为使用的数据不同，论文里发表的结果无法互相比较。

马库斯比很多同行更早地发现建立标准语料库在自然语言处理研究中的重要性。于是，马库斯利用自己的影响力，让美国自然科学基金会（National Science Foundation，简称 NSF）和 DARPA 出资立项，联络了多所大学和研究机构，建立了数百个标准的语料库组织（Linguistic Data Consortium，简称 LDC）。其中最著名的语料库是 Penn Tree Bank[1]。Penn Tree Bank起初是收集了一些真实的书面英语(《华尔街日报》)语句，人工进行词性标注和语法树构建等，作为全世界自然语言处理学者研究和实验的统一语料库。后来，由于它被全世界认可，美国自然科学基金会不断追加投入，建立起了覆盖多种语言（包括中文）的语料库。对每一种语言，它有几十万到几百万字的有代表性的句子，每个句子都有词性标注、语法分析树等。LDC 后来又建立了语音、机器翻译等很多数据库，为全世界自然语言处理科学家共享。如今，在自然语言处理方面发表论文，几乎都要提供基于 LDC 语料库的测试结果。

1 因为宾夕法尼亚大学又简称 UPenn，或者 Penn，这个由该大学领头建立的数据库便被命名为 Penn Tree Bank。Tree Bank 直接的意思是（语法）树的银行，寓意大量的语法树。

在过去 20 年机器学习和自然语言处理领域，80% 的成果来自于数据量的增加。马库斯对这些领域数据的贡献可以说是独一无二的。当然，凭借对数据的贡献，还不足以让马库斯获得教父的地位。马库斯有点像日本围棋领域的木谷实[2]，他的影响力很大程度上是靠他的弟子传播出去的。

2 日本著名的围棋教育家，他的弟子石田方夫、加藤正夫、武宫正树、小林光一和赵治勋在 1970-2000 年统治日本棋坛 30 年。

给予他的博士生研究自己感兴趣的课题的自由，这是他之所以桃李满天下的原因。马库斯的博士生研究的题目覆盖了自然语言处理的很多领域，而且题目之间几乎没有相关性，因为这些题目大多是博士生自己找的，而不是马库斯指定的。他的做法和中国大部分博士生导师完全不同。马库斯对几乎所有的自然语言处理领域都有独到的见解，马库斯让博士生提出自己有兴趣的课题，或者用他已有的经费支持学生，或者去为他们的项目申请经费。马库斯高屋建瓴，能够很快地判断一个研究方向是否正确，省去了博士生很多做无谓尝试（Try-And-Error）的时间。因此他的博士毕业生质量非常高，而且有些很快就拿到了博士学位。

由于马库斯宽松的管理方式，他培养的博士生在研究和生活上都是个性迥异。有些人善于找到间接快速的方法和容易做出成绩的题目，有的人习惯啃硬骨头；有些人三四年就拿到博士去当教授了，而有些人“赖在”学校里七八年不走，最后出一篇高质量的博士论文。这些各有特点的年轻学者，后来分别能适应文化迥异的各个大学和公司。

马库斯教授长期担任宾夕法尼亚大学计算机系主任，直到 2002 年从 AT&T 实验室找到费尔南多·皮耶尔替代他为止。作为一个管理者，马库斯在专业设置方面显示出远见卓识，马库斯将宾夕法尼亚大学规模很小的计算机系发展成在学术界具有盛名和影响力的强系。在世界各种大学研究生院的排名中，一般来讲，规模大的院系比规模小的要占不少便宜，因为前者学科齐全。马库斯的风格是把一个系变强而不是变大。马库斯在几年前互联网很热门、很多大学开始进行互联网研究时，看到生物信息学（Bioinformatics）的重要性，在宾夕法尼亚大学设立这个专业，并且在其他大学还没有意识到时，开始招聘这方面的教授。马库斯还建议

一些相关领域的教授，包括后来的系主任皮耶尔把一部分精力转到生物信息学方面。等到网络泡沫破裂以后，很多大学的计算机系开始向生物信息学转向，但是发现已经很难找到这个领域好的教授了。

我有幸和他同在约翰·霍普金斯大学计算机系的顾问委员会任职多年，每年有两次在一起讨论计算机系的研究方向。和宾夕法尼亚大学类似，约翰·霍普金斯大学的计算机系也很小，发展也面临同样的问题。马库斯的主张一贯是建立几个世界上最好的专业，而不是专业最齐全的系。我觉得，当今中国的大学，最需要的就是马库斯这样卓有远见的管理者。

2 从宾夕法尼亚大学走出的精英们

当今自然语言处理领域年轻一代的世界级专家，相当大一部分来自宾夕法尼亚大学马库斯的实验室。他们为人做事风格迥异，共同的特点是年轻有为。这里介绍其中两人，迈克尔·柯林斯和艾里克·布莱尔，因为他们代表两种截然不同的风格。

2.1 柯林斯：追求完美

我在“数学之美”系列中一直强调一个好方法在形式上应该是简单的。但是，事实上，自然语言处理中也有一些特例，比如有些学者将一个问题研究到极致，执著追求完善甚至可以说达到完美的程度。他们的工作对同行有很大的参考价值，因此在科研中很需要这样的学者。在自然语言处理方面新一代的顶级人物迈克尔·柯林斯就是这样的人。

柯林斯1993年从剑桥大学硕士毕业后，师从自然语言处理大师马库斯，用五年多时间完成了博士论文，从宾夕法尼亚大学获得博士学位。在他的师兄弟中，他比用三年拿到博士学位的雅让斯基要长很多，但是比赖在学校不走的恩斯勒（Jason Eisner）要快不少。但无论是比他花的时间长的，还是短的，论文做得都没有柯林斯好。

在做博士期间，柯林斯写了一个后来以他名字命名的自然语言文法分析器（Sentence Parser），可以对书面语的每一句话进行准确的文法分析。前面提到，文法分析被认为是很多自然语言应用的基础。柯林斯的师兄布莱尔和拉纳帕提以及师弟恩斯勒都完成了相当不错的语言文法分析器，照理讲，柯林斯不应该再选择这个课题了，因为很难出成果。但柯林斯却是一个要把技术潜力挖掘到极致的人，他在这方面的追求很像乔布斯在产品上的追求。他的师兄弟选择这个题目都是为了验证自己的理论：布莱尔是为了证明他的“基于变换”的机器学习方法的有效性，拉纳帕提是为了证明最大熵模型，恩斯勒是为了证明有限状态机。柯林斯和他的师兄弟不同，他做文法分析器的出发点不是为了验证一个理论，而是要做一个世界上最好的分析器。

在这样的思想指导下，柯林斯做出了在相当长一段时间内世界上最好的文法分析器。柯林斯成功的关键在于将文法分析的每一个细节都研究得很仔细。柯林斯用的数学模型也很漂亮，整个工作可以用完美来形容。我曾因为研究的需要，找柯林斯要过他文法分析器的源程序，他很爽快地给了我。我试图将他的程序修改一下来满足我特定应用的要求，但后来发现，他的程序细节太多，以至于很难进一步优化。柯林斯不是成功做出文法分析器的第一人，甚至不是第二、第三人。但是从某种程度上讲可能是最后一人，在过去的七八年里，他还在这个领域不断改进，不断突破，大有其他科学家从此不必再做文法分析器的架势！

柯林斯的博士论文堪称自然语言处理领域的范文。它像一本优秀的小说，把所有事情的来龙去脉介绍得清清楚楚，任何有一点计算机和自然语言处理知识的人，都可以轻而易举读懂他复杂的方法。

柯林斯毕业后，在 AT&T 实验室度过了三年快乐的时光。在那里柯林斯完成了许多世界一流的研究工作，诸如隐含马尔可夫模型的区别性训练方法，卷积核在自然语言处理中的应用，等等。三年后，AT&T 停止了自然语言处理方面的研究，柯林斯幸运地在麻省理工学院找到教职。在

麻省理工学院的短短七年间，柯林斯三次获得 EMNLP 最佳论文奖，两次获得 UAI 最佳论文奖和一次 CoNLL 最佳论文奖。一般一个一流的科学家，一生也就获得两三次最佳论文奖，而柯林斯把获奖当成了家常便饭！相比其他同行，这种成就是世界上独一无二的。柯林斯的特点就是把事情做到极致。如果说有人喜欢"繁琐哲学"，柯林斯就是一个。

柯林斯在麻省理工学院获得了终身教职后，2011 年被哥伦比亚大学以潘迪特（Vikram Pandit）[3] 教席的教授职位 [4] 挖走。

3 花旗银行的 CEO。

4 欧美一些大学设有以私人或者机构命名的教席，授予著名的教授。比如剑桥大学著名的卢卡斯数学教席（Lucasian Mathematics Professor）曾经授予牛顿、狄拉克等著名科学家，当代著名物理学家霍金也曾担任此教席。

2.2 布莱尔：简单才美

在研究方法上，站在柯林斯对立面的典型是他的师兄艾里克·布莱尔、拉纳帕提和雅让斯基等，后面两人在前面章节中已经介绍过了。与柯林斯从工业界到学术界相反，布莱尔的职业路径是从学术界走到工业界。前者在学术界换了大学，后者到了工业界后也换过公司。与柯林斯的研究方法相反，布莱尔总是试图寻找简单得不能再简单的方法。但相同的是，二人的职位都越做越高。布莱尔的成名作是基于变换规则的机器学习方法（Transformation Rule Based Machine Learning）。这个方法名称虽然很复杂，其实非常简单。以拼音转汉字为例来说明它：

第一步，把每个拼音对应的汉字中最常见的找出来作为第一遍变换的结果，当然结果有不少错误。比如，"常识"可能被转换成"长识"；

第二步，可以说是"去伪存真"，用计算机根据上下文，列举所有的同音字替换的规则，比如，如果 chang 被标识成"长"，但是后面的汉字是"识"，则将"长"改成"常"；

第三步，应该就是"去粗取精"，将所有的规则应用到事先标识好的语料中，挑出有用的，删掉无用的。然后重复二三步，直到找不出有用的为止。

布莱尔就靠这么简单的方法，在很多自然语言研究领域，取得了几乎最好的结果。由于他的方法再简单不过了，许许多多的人都跟着学。布莱

尔可以算是我在美国的第一个业师，我们俩就用这么简单的方法作词性标注（Part of Speech Tagging），也就是把句子中的词标成名词动词，很多年内无人能超越。（最后超越我们的是后来加入 Google 的一名荷兰工程师，用的是同样的方法，但是做得细致很多。）布莱尔离开学术界后去了微软研究院。在那里的第一年，他一人一年完成的工作比组里其他所有人许多年做的工作的总和还多。后来，布莱尔又加入了一个新组，依然是高产科学家。据说，他的工作真正被微软重视要感谢 Google，因为有了 Google，微软才对他从人力物力上给予了巨大的支持，使得布莱尔成为微软搜索研究的领军人物之一。在研究方面，布莱尔有时不一定能马上知道应该怎么做，但是能马上否定掉一种不可能的方案。这和他追求简单的研究方法有关，他能在短时间内大致摸清每种方法的好坏。

如果说柯林斯是个“务于精纯”的精深专才，布莱尔则更像“观其大略”的通才。在微软研究院，布莱尔创建了数据挖掘和搜索研究领域，并在此基础上将它变为微软互联网研究中心（Internet Service Research Center），后来又作为总经理管理着微软的广告实验室（AdCenter Lab）。2009 年布莱尔离开微软到 eBay，出任 CTO 兼主管研究的副总裁。

布莱尔总是善于寻找简单却有效的方法，而又从不隐瞒自己的方法，所以他总是很容易被包括我在内的很多人赶上和超过。好在布莱尔对此毫不介意，而且很喜欢别人追赶他，因为，当人们在一个研究方向上超过他时，说明他开创的领域有意义，同时他已经调转船头驶向其他方向了。2005 年 Google 上市后，微软对搜索的投入加大力度，我和他的位置调换了一下，因为微软成了 Google 的追赶者。当我告诉他我们的位置调了个个儿时，布莱尔对我说，有一件事我永远追不上他，那就是他比我先有了第二个孩子。

第23章 布隆过滤器

1 布隆过滤器的原理

在日常生活中，包括设计计算机软件时，经常要判断一个元素是否在一个集合中。比如，在字处理软件中，需要检查一个英语单词是否拼写正确（也就是要判断它是否在已知的字典中）；在 FBI，一个嫌疑人的名字是否已经在嫌疑名单上；在网络爬虫里，一个网址是否已访问过，等等。最直接的方法就是将集合中全部的元素存在计算机中，遇到一个新元素时，将它和集合中的元素直接比较即可。一般来讲，计算机中的集合是用哈希表（Hash Table）来存储的。它的好处是快速准确，缺点是耗费存储空间。当集合比较小时，这个问题不明显，当集合规模巨大时，哈希表存储效率低的问题就显现出来了。比如，一个像 Yahoo、Hotmail 和 Gmail 那样的公众电子邮件（Email）提供商，总是需要过滤来自发送垃圾邮件的人（Spamer）的垃圾邮件。采用的一个办法就是记录下那些发垃圾邮件的 Email 地址。由于那些发送者不停地在注册新的地址，全世界少说也有几十亿个发垃圾邮件的地址，将这些地址都存起来，需要大量的网络服务器。如果用哈希表，每存储一亿个 Email 地址，就需要 1.6GB 的内存（用哈希表实现的具体办法是将每一个 Email 地址对应成一个 8 字节的信息指纹，然后将这些信息指纹存入哈希表，由于哈希表的存储效率一般只有 50%，因此一个 Email 地址需要占用 16 个字节。一亿个地址大约

要 1.6GB，即 16 亿字节的内存）。因此，存储几十亿个邮件地址可能需要上百 GB 的内存。除非是超级计算机，一般服务器是无法存储的。

今天，我们介绍一种称作布隆过滤器的数学工具，它只需要哈希表 1/8 到 1/4 的大小就能解决同样的问题。

布隆过滤器是由伯顿·布隆（Burton Bloom）于 1970 年提出的。它实际上是一个很长的二进制向量和一系列随机映射函数。我们通过上面的例子来说明其工作原理。

假定存储一亿个电子邮件地址，先建立一个 16 亿二进制（比特），即两亿字节的向量，然后将这 16 亿个二进制位全部清零。对于每一个电子邮件地址 X，用 8 个不同的随机数产生器 $(F_1,F_2,\dots,F_8)$ 产生 8 个信息指纹 $(f_1,f_2,\dots,f_8)$。再用一个随机数产生器 G 把这 8 个信息指纹映射到 1-16 亿中的 8 个自然数 $g_1,g_2,\dots,g_8$。现在把这 8 个位置的二进制全部设置为 1。对这一亿个电子邮件地址都进行这样的处理后，一个针对这些电子邮件地址的布隆过滤器就建成了，见下图。

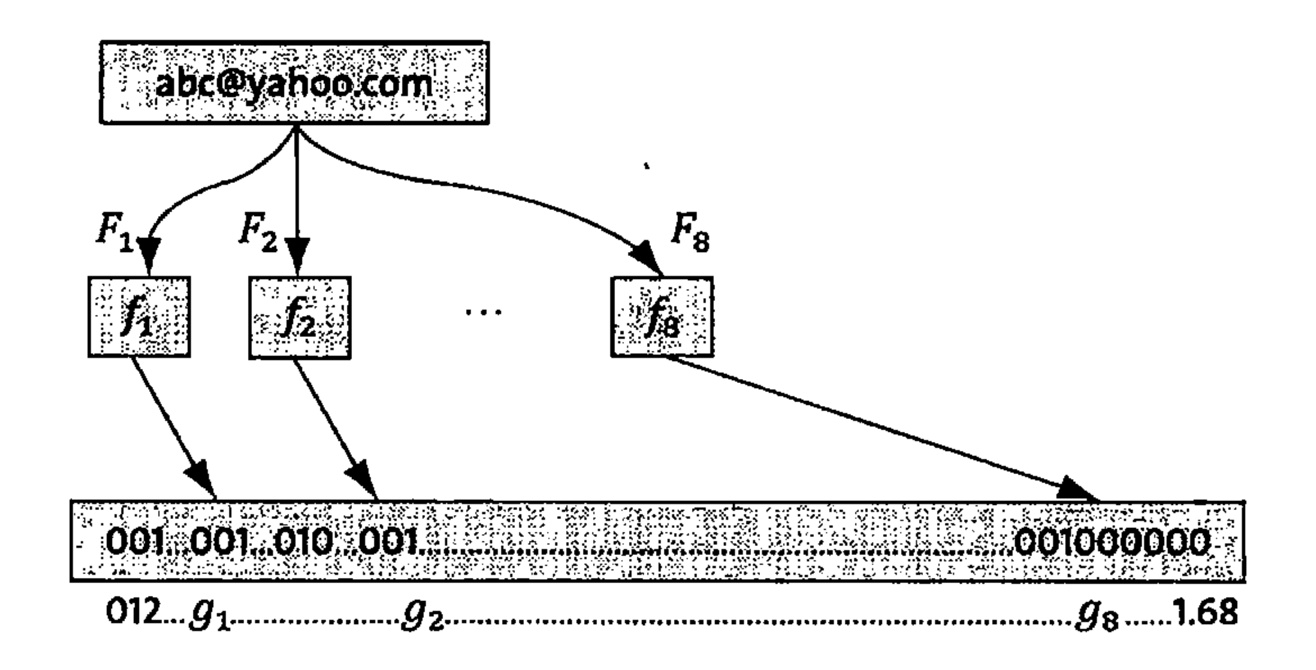


图 23.1　布隆过滤器的映射方法

现在，让我们看看如何用布隆过滤器来检测一个可疑的电子邮件地址 Y 是否在黑名单中。用相同的 8 个随机数产生器 $(F_1,F_2,\dots,F_8)$ 对这个地址产生 8 个信息指纹 $s_1,s_2,\dots,s_8$，然后将这 8 个指纹对应到布隆过滤器的 8 个二进制位，分别是 $t_1,t_2,\dots,t_8$。如果 Y 在黑名单中，显然，$t_1,t_2,\dots,t_8$ 对应的

8 个二进制数一定是 1。这样，再遇到任何在黑名单中的电子邮件地址，都能准确地发现。

布隆过滤器决不会漏掉黑名单中的任何一个可疑地址。但是，它有一个不足之处。也就是它有极小的可能将一个不在黑名单中的电子邮件地址也判定为在黑名单中，因为有可能某个好的邮件地址在布隆过滤器中对应的 8 个位置“恰巧”被（其他地址）设置成 1。好在这种可能性很小。我们把它称为误识别率。在上面的例子中，误识别率在万分之一以下。误识别的理论分析会在延伸阅读中介绍。

布隆过滤器的好处在于快速、省空间，但是有一定的误识别率。常见的补救办法是再建立一个小的白名单，存储那些可能被误判的邮件地址。

2 延伸阅读：布隆过滤器的误识别问题

读者背景知识：概率论。

上一节中提到，布隆过滤器的一个不足之处就是它可能把不在集合中的元素错判成集合中的元素，这在检验上被称为“假阳性”。这个概率很小，但是究竟有多小，是否可以忽略。

估算假阳性的概率并不难。假定布隆过滤器有 m 比特，里面有 n 个元素，每个元素对应 k 个信息指纹的哈希函数，当然这 m 比特里有些是 1，有些是 0。先来看看某个比特为零的概率。比如，在这个布隆过滤器中插入一个元素，它的第一个哈希函数会把过滤器中的某个比特置成 1，因此，任何一个比特被置成 1 的概率是 $1/m$，它依然是 0 的概率则是 $1-\frac{1}{m}$。

对于过滤器中一个特定的位置，如果这个元素的 k 个哈希函数都没有把它设置成 1，其概率是 $\left(1-\frac{1}{m}\right)^k$。如果过滤器中插入的第二个元素，某个特定的位置依然没有被设置成 1，其概率为 $\left(1-\frac{1}{m}\right)^{2k}$。如果插入了 n 个元

素还没有把某个位置设置成 1，其概率是$\left(1-\frac{1}{m}\right)^{kn}$。反过来，一个比特在插入了$n$个元素后，被置成 1 的概率则是$1-\left(1-\frac{1}{m}\right)^{kn}$。

现在假定这n个元素都放到布隆过滤器中了，新来一个不在集合中的元素，由于它的信息指纹的哈希函数都是随机的，因此，它的第一个哈希函数正好命中某个值为 1 的比特的概率就是上述概率。一个不在集合中的元素被误识别为在集合中，需要所有的哈希函数对应的比特值均为 1，其概率为

$$\left(1-\left[1-\frac{1}{m}\right]^{kn}\right)^{k} \approx \left(1-e^{-\frac{kn}{m}}\right)^{k} \quad (23.1)$$

化简后为

$$\mathrm{p}=\left(1-e^{-\frac{\left(\frac{m}{n}\ln 2\right)n}{m}}\right)^{\left(\frac{m}{n}\ln 2\right)} \quad (23.2)$$

如果n比较大，可以近似为

$$(1-e^{-k(n+0.5)/(m-1)})^{k} \approx \ (1-e^{-\frac{kn}{m}})^{k} \quad (23.3)$$

假定一个元素用 16 比特，$k=8$，那么假阳性的概率是万分之五。在大部分应用中是可以忍受的。下表是不同的m/n比例，以及k情况下的假阳性概率（下表由原麦迪逊威斯康星大学曹培（Pei Cao）教授提供，她目前在 Facebook 任职）。

表 23.1 不同 m/n 和 k 情况下，布隆过滤器的误识别概率
（数据来源：http://pages.cs.wisc.edu/~cao/papers/summary-cache/node8.html）

m/n	k	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
2	1.39	0.393	0.400						
3	2.08	0.283	0.237	0.253					
4	2.77	0.221	0.155	0.147	0.160				
5	3.46	0.181	0.109	0.092	0.092	0.101			
6	4.16	0.154	0.0804	0.0609	0.0561	0.0578	0.0638		
7	4.85	0.133	0.0618	0.0423	0.0359	0.0347	0.0364		

续表

m/n	k	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
8	5.55	0.118	0.0489	0.0306	0.024	0.0217	0.0216	0.0229	
9	6.24	0.105	0.0397	0.0228	0.0166	0.0141	0.0133	0.0135	0.0145
10	6.93	0.0952	0.0329	0.0174	0.0118	0.00943	0.00844	0.00819	0.00846
11	7.62	0.0869	0.0276	0.0136	0.00864	0.0065	0.00552	0.00513	0.00509
12	8.32	0.08	0.0236	0.0108	0.00646	0.00459	0.00371	0.00329	0.00314
13	9.01	0.074	0.0203	0.00875	0.00492	0.00332	0.00255	0.00217	0.00199
14	9.7	0.0689	0.0177	0.00718	0.00381	0.00244	0.00179	0.00146	0.00129
15	10.4	0.0645	0.0156	0.00596	0.003	0.00183	0.00128	0.001	0.000852
16	11.1	0.0606	0.0138	0.005	0.00239	0.00139	0.000935	0.000702	0.000574
17	11.8	0.0571	0.0123	0.00423	0.00193	0.00107	0.000692	0.000499	0.000394
18	12.5	0.054	0.0111	0.00362	0.00158	0.000839	0.000519	0.00036	0.000275
19	13.2	0.0513	0.00998	0.00312	0.0013	0.000663	0.000394	0.000264	0.000194
20	13.9	0.0488	0.00906	0.0027	0.00108	0.00053	0.000303	0.000196	0.00014
21	14.6	0.0465	0.00825	0.00236	0.000905	0.000427	0.000236	0.000147	0.000101
22	15.2	0.0444	0.00755	0.00207	0.000764	0.000347	0.000185	0.000112	7.46e-05
23	15.9	0.0425	0.00694	0.00183	0.000649	0.000285	0.000147	8.56e-05	5.55e-05
24	16.6	0.0408	0.00639	0.00162	0.000555	0.000235	0.000117	6.63e-05	4.17e-05
25	17.3	0.0392	0.00591	0.00145	0.000478	0.000196	9.44e-05	5.18e-05	3.16e-05
26	18	0.0377	0.00548	0.00129	0.000413	0.000164	7.66e-05	4.08e-05	2.42e-05
27	18.7	0.0364	0.0051	0.00116	0.000359	0.000138	6.26e-05	3.24e-05	1.87e-05
28	19.4	0.0351	0.00475	0.00105	0.000314	0.000117	5.15e-05	2.59e-05	1.46e-05
29	20.1	0.0339	0.00444	0.000949	0.000276	9.96e-05	4.26e-05	2.09e-05	1.14e-05
30	20.8	0.0328	0.00416	0.000862	0.000243	8.53e-05	3.55e-05	1.69e-05	9.01e-06
31	21.5	0.0317	0.0039	0.000785	0.000215	7.33e-05	2.97e-05	1.38e-05	7.16e-06
32	22.2	0.0308	0.00367	0.000717	0.000191	6.33e-05	2.5e-05	1.13e-05	5.73e-06

3 小结

布隆过滤器背后的数学原理在于两个完全随机的数字冲突的概率很小，因此，可以在很小的误识别率条件下，用很少的空间存储大量信息。常见的补救误识别的办法是再在建立一个小的白名单，存储那些可能被别

误判的信息。由于布隆过滤器中只有简单的算术运算，因此它的速度很快，使用方便。

第24章 马尔可夫链的扩展 —— 贝叶斯网络

1 贝叶斯网络

前面的章节中多次提到马尔可夫链（Markov Chain），它描述了一种状态序列，其每个状态值取决于前面有限个状态。这种模型，对很多实际问题来讲是一种很粗略的简化。在现实生活中，很多事物相互的关系并不能用一条链来串起来，它们之间的关系可能是交叉的、错综复杂的。比如在下图中可以看到，心血管疾病和它的成因之间的关系是错综复杂的。显然无法用一个链来表示（图 24.1）。

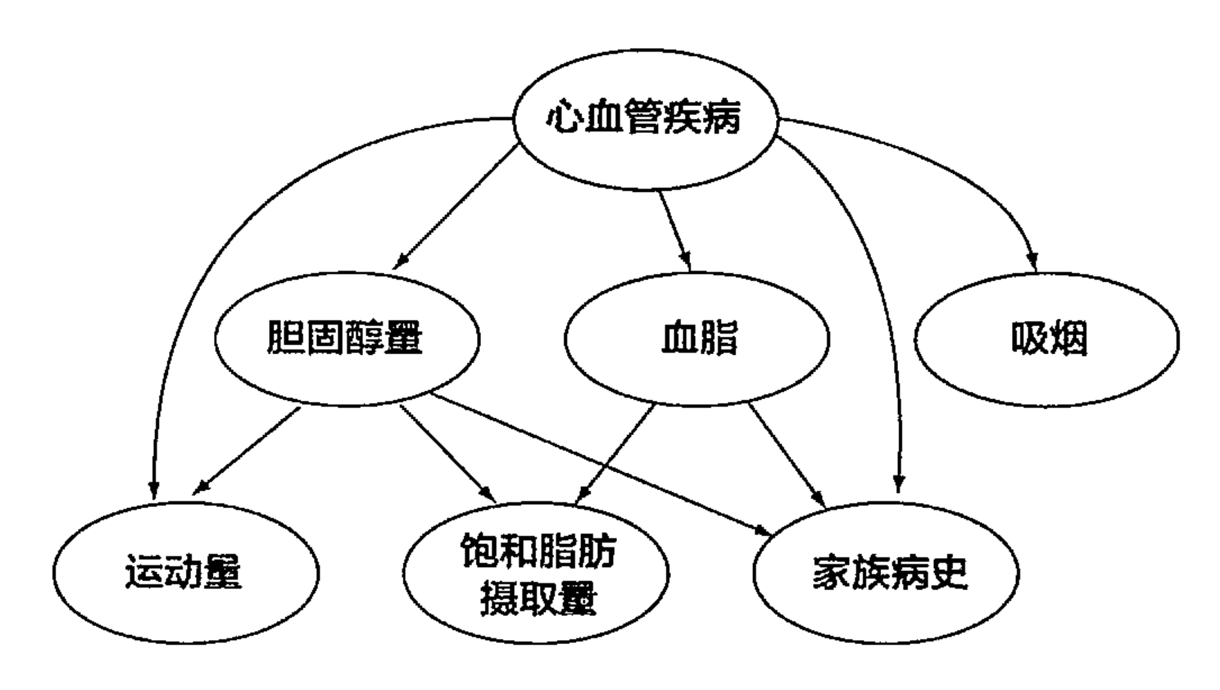


图 24.1 描述心血管疾病和成因的一个简单的贝叶斯网络

可以把上述的有向图看成一个网络，其中每个圆圈表示一个状态。状态之间的连线表示它们的因果关系。比如从心血管疾病出发到吸烟的弧线表示心血管疾病可能和吸烟有关。假定在这个图中马尔可夫假设成立，即每一个状态只和与它直接相连的状态有关，而和它间接相连的状态没有直接关系，那么它就是贝叶斯网络。不过要注意，两个状态A和B之间没有直接的有向弧链接，只说明它们之间没有直接的因果关系。但这并不表明状态A不会通过其他状态间接地影响状态B，只要在图中有一条从A到B的路径，这两个状态就还是有间接的相关性的。所有这些（因果）关系，都可以有一个量化的可信度（Belief），用一个概率描述。也就是说，贝叶斯网络的弧上可以有附加的权重。马尔可夫假设保证了贝叶斯网络便于计算。我们可以通过这样一张网络估计出一个人患心血管疾病的可能性。

在网络中每个节点概率的计算，都可以用贝叶斯公式来进行，贝叶斯网络因此而得名。由于网络的每个弧都有一个可信度，贝叶斯网络也被称作信念网络（Belief Networks）。在上面的例子中，我们做一个简化，假定只有三个状态“心血管疾病”、“高血脂”和“家族病史”。用这个简化的例子来说明这个网络中的一些概率是如何通过贝叶斯公式计算出来的。为了简单起见，假定每个状态取值只有“有、无”两种，如图24.2 所示。图中的三个表各自代表了每个状态和组合状态不同取值时的条件概率，比如左上角的表说明，如果有家族病史时，高血脂的可能性是 0.4，如果无家族病史，这个可能性只有 0.1。

心血管疾病 \ 家族病史，高血脂	有	无
有，有	0.9	0.1
有，无	0.4	0.6
有，有	0.4	0.6
有，无	0.1	0.9

高血脂 \ 家族病史	有	无
有	0.4	0.6
无	0.1	0.9

有	无
0.2	0.8

图 24.2 一个简化的贝叶斯网络

如果要计算“家族病史”、“高血脂”和“心血管疾病”三者的联合概率分布，可以利用贝叶斯公式：

P（家族病史，高血脂，心血管疾病）= P（心血管疾病 | 家族病史，高血脂）× P（高血脂 | 家族病史）× P（家族病史） （24.1）

只要代入表中的数值就能计算出概率。

再比如，如果问心血管疾病有多大的可能是由家族病史引起的，也可以通过这个贝叶斯网络计算出来。

P（有家族病史 | 有心脏病）= P（有家族病史，有心脏病）/ P（有心脏病） （24.2）

其中：

P（有家族病史，有心脏病）= P（有家族病史，有心脏病，无高血脂）+ P（有家族病史，有心脏病，有高血脂） （24.3）

P（有心脏病）= P（有家族病史，有心脏病，无高血脂）+ P（有家族病史，有心脏病，有高血脂）+ P（无家族病史，有心脏病，

无高血脂）+ P（无家族病史，有心脏病，有高血脂）（24.4）

上面两个式子中的每一项都可以通过（24.1）计算出来，代人图 24.2 中的值，这个概率为：

P（有家族病史，有心脏病）= 0.18 × 0.4 + 0.12 × 0.4
= 0.12

P（有心脏病）= 0.12 + 0.08 × 0.4 + 0.72 × 0.1
= 0.12 + 0.144
= 0.264

将结果代人（24.2），得到：P（有家族病史 | 有心脏病）= 45%

因此，虽然有家族病史的人只占人口的 20%，但是他们占了有心血管疾病人数的 45%，发病率远远高于没有家族病史的人。

我们在计算上面的贝叶斯网络中每个状态的取值时，只考虑了前面一个状态，这一点和马尔可夫链相同。但是，贝叶斯网络的拓扑结构比马尔可夫链灵活，它不受马尔可夫链的链状结构的约束，因此可以更准确地描述事件之间的相关性。可以讲，马尔可夫链是贝叶斯网络的特例，而贝叶斯网络是马尔可夫链的推广。

使用贝叶斯网络必须先确定这个网络的拓扑结构，然后还要知道各个状态之间相关的概率，也就是图 24.2 中的那些表。得到拓扑结构和这些参数的过程分别叫做结构训练和参数训练，统称训练。和训练马尔可夫模型一样，训练贝叶斯网络要用一些已知的数据。比如训练上面的网络，需要知道一些心血管疾病和吸烟、家族病史等有关的情况。相比马尔可夫链，贝叶斯网络的训练比较复杂，从理论上讲，它是一个 NP 完备问题（NP-Complete），也就是说，对于现在的计算机是不可计算的。但是，对于某些应用，这个训练过程可以简化，并在计算机上实现。

值得一提的是 IBM 华生实验室的茨威格博士（Geoffrey Zweig）和西雅图华盛顿大学的比尔默（Jeff Bilmer）教授完成了一个通用的贝叶斯网络的工具包，提供给对贝叶斯网络有兴趣的研究者。

贝叶斯网络在图像处理、文字处理、支持决策等方面有很多应用。在文字处理方面，语义相近的词之间的关系可以用一个贝叶斯网络来描述。我们利用贝叶斯网络，可以找出近义词和相关的词，因而在 Google 搜索和 Google 广告中都有直接的应用。

2　贝叶斯网络在词分类中的应用

可以用基于统计的模型分析文本，从中抽取概念，分析主题。不妨把这样的模型称为主题模型（Topic Model）。前面章节中提到的从一篇文章里得到它的特征向量，然后把这个特征向量通过余弦相似性（距离）对应到一个主题的特征向量，便是统计主题模型的一种。这里介绍通过贝叶斯网络建立的另一种模型，Google 的 Rephil。由于不便透露太多 Rephil 的细节，因此在这里我改用自己的语言讲述它的原理，听过这个主题报告或者读过报告幻灯片的读者可能会发现我的讲法略有不同，但原理是相同的。

在介绍 Rephil 以前，我们先来回顾一下文本分类。假如我们有一亿篇文本，我们或者使用（文本和关键词）关联矩阵的奇异值分解，或者使用余弦距离的聚类，可以将它们分成一万类，当然这个类别的数量可以由工程人员自己定。对于一篇文章，可以把它归到一类或者若干类中。同一类的文章，共享很多关键词。至于关键词是怎么产生的，可以有各种办法，比如人工挑选。但是大规模数据处理需要用自动的办法。这样，不同的文章通过关键词建立了一种关系，这种关系表明一些文章属于同一类或者不属于同一类。

如果把文本和关键词的关联矩阵转 90 度，进行奇异值分解，或者对每个词以文本作为维度，建立一个向量，再进行向量的聚类，那么得到的是对

词的一个分类而不是对文本的分类。分出来的每一类我们称为一个概念。

显然，一个概念可以包含多个词，一个词也可以属于多个概念。类似地，一篇文章可以对应多个概念，而一个概念也对应多篇文章。现在可以用贝叶斯网络建立一个文章、概念和关键词之间的联系：

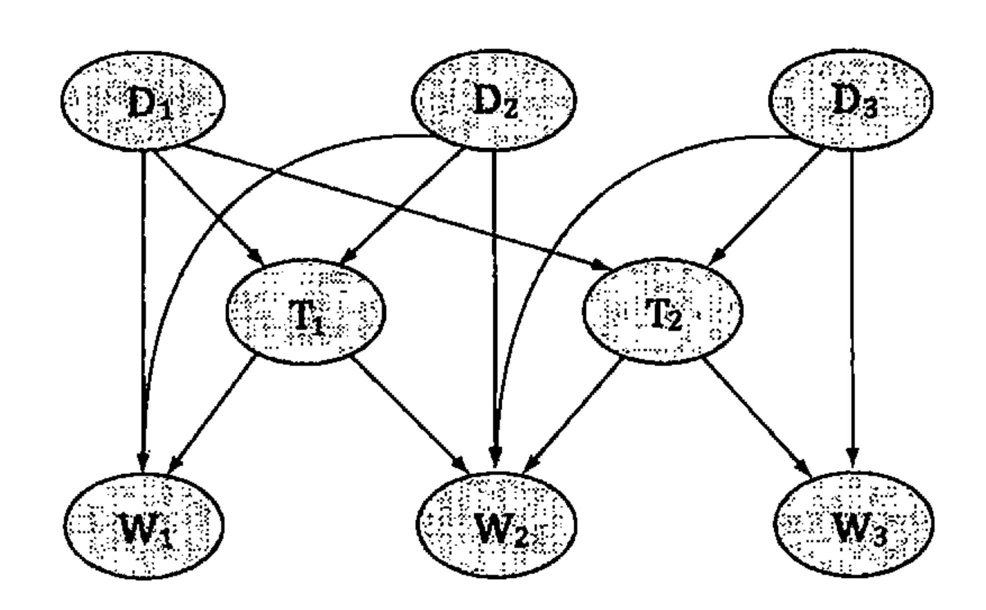


图 24.3　描述文章（D）、主题（T）和关键词（W）的贝叶斯网络

在图 24.3 中，文章和关键词本身有直接的关联，它们两者都还和概念有直接关联，同时它们通过主题还有间接的关联。就如同 24.1 节中的那个例子，可以找到心血管疾病和家族病史之间的关系一样，也可以通过上面这个网络找到每个概念和每个词之间的相关性，以及词和词之间的相关性。只是这个相关性未必是条件概率。

2002 年，Google 的工程师们利用贝叶斯网络，建立了文章、概念和关键词的联系，将上百万关键词聚合成若干概念的聚类，称为 Phil Cluster。这项工作最初甚至没有考虑应用背景，只是觉得概念的提取将来对信息处理会有帮助。而最早的应用是广告的扩展匹配，这是 Phil Cluster 完成后几个月里的事情。由于早期只考虑了关键词和文本的关系，对关键词上下文关系考虑较少，因此这个概念的聚类过于广泛，比如计算机和股票可能被聚在一类中，汽车和美国总统的名字聚在一起。这严重影响了 Google 各个项目组对 Phil Cluster 的接受程度。2004 年，Google 开始重构 Phil Cluster，这次采用了比原先多几百倍的数据，考虑的关键词的相似性也从原来的在文本中同现扩展为在上下文中同现，同时支持不同颗

粒的概念。因为是重构，所以项目的名称改为 Rephil。Rephil 聚合了大约 1 200 万个词，将它们聚合到上百万的概念中。一个概念一般有十几个到上百个词。由于 Rephil 的质量比以前有很大提高，因此从广告到搜索都用到了它的成果。久而久之，就没有多少人知道 Phil 了。Rephil 的网络结构比原先的 Phil 复杂很多，但是原理类似。

3　延伸阅读：贝叶斯网络的训练

读者背景知识：概率论。

使用贝叶斯网络首先要确定它的结构。对于简单的问题，比如上面那个心血管疾病的例子，由专家直接给出它的结构即可。但是，稍微复杂一点的问题已经无法人工给出结构了，需要通过机器学习得到。

优化的贝叶斯网络结构要保证它产生的序列（比如"家族病史→高血脂→心血管疾病"就是一个序列）从头走到尾的可能性最大，如果是使用概率做度量，那就是后验概率最大。当然，产生一个序列可以有多条路径，从理论上讲，需要完备的搜索（Exhaustive Search），即考虑每一条路径，才能得到全局最优。但是这样的计算复杂度是 NP-Hard（详见附录一），因此一般采用贪婪的算法（Greedy Algorithm），也就是在每一步时，沿着箭头的方向寻找有限步。当然这样会导致陷入局部最优，并且最终远离全局最优解。一个防止显然局部最优的方法，就是采用蒙特卡罗（Monte Carlo）的方法，用许多随机数在贝叶斯网络中试一试，看看是否显然局部最优。这个方法的计算量比较大。最近，新的方法是利用信息论，计算节点之间两两的互信息，然后只保留互信息较大的节点直接的连接，然后再对简化了的网络进行完备的搜索，找到全局优化的结构。

确定贝叶斯网络的结构后，就要确定这些节点之间的弧的权重了，假定这些权重用条件概率来度量。为此，需要一些训练数据，我们需要做的就是优化贝叶斯网络的参数，使得观察到的这些数据的概率（即后验概率）$P(D|\theta)$ 达到最大。这个过程就是前面介绍过的 EM 过程（Expectation-

Maximization Process）。

在计算后验概率时，计算的是条件X和结果Y之间的联合概率$P(X,Y)$。我们的训练数据会提供一些$P(X,Y)$之间的限制条件，而训练出来的模型要满足这些限制条件。回顾我们在“最大熵模型”一章中介绍的内容，这个模型应该是满足给定条件的最大熵模型。因此，涉及到最大熵模型的训练方法在这里都可以使用。

最后，需要指明结构的训练和参数的训练通常是交替进行的。也就是先优化参数，再优化结构，然后再次优化参数，直至得到收敛或者误差足够小的模型。

对贝叶斯网络有特别兴趣的读者，可以阅读比尔默和茨威格共同发表的论文：

http://ssli.ee.washington.edu/~bilmes/pgs/sort_date.html

4 小结

从数学的层面讲，贝叶斯网络是一个加权的有向图，是马尔可夫链的扩展。而从认识论的层面看：贝叶斯网络克服了马尔可夫链那种机械的线性的约束，它可以把任何有关联的事件统一到它的框架下面。因此，贝叶斯网络有很多应用，除了我介绍的文本分类和概念抽取外，在生物统计、图像处理、决策支持系统和博弈论中都有广泛应用。贝叶斯网络的描述非常简单易懂，但导出的模型却非常复杂。

第25章 条件随机场和句法分析

条件随机场是计算联合概率分布的有效模型，而句法分析似乎是英文课上英语老师教的东西，这两者有何联系呢？我们先从句法分析的演变谈起。

1 句法分析计算机算法的演变

我们讲到自然语言的句法分析就是为每个句子建立一棵语法树。以前受形式语言学的影响，采用基于规则的方法，那么建这棵语法树的过程就是不断地使用规则将树的末端节点逐级向上合并，直到合并出根节点，即一个整句。这个方法是自底向上的，当然也可以自顶向下进行。不论是哪一种，都有一个无法避免的问题，就是选择规则时不可能一次选对。如果在某一步一旦走岔路了，需要回溯很多步。因此，这两种方法的计算复杂度都大得不得了，也不可能分析复杂的句子。

上个世纪 80 年代以后，布朗大学计算机系的计算语言学家尤金·查尼阿克（Eugene Charniack）统计出文法规则的概率，在选择文法规则时，坚持一个原则 —— 让被分析的句子的语法树概率达到最大。这个看似非常简单直接的方法，使得句法分析的计算量一下降低了很多，而准确性却大大提高。查尼阿克无形中在数学和句法分析上搭建了一个桥梁。而搭建这个桥梁的第二个人就是马库斯的学生拉纳帕提。

用马库斯的另一个高足布莱尔的话讲，拉纳帕提是绝顶聪明的人，不过从我个人接触来看，我认为他更大的长处在于极强的动手能力。拉纳帕提从全新的角度来看待句法分析问题 —— 把句法分析看成是一个括括号的过程。

还是举前面章节中的那个很长的句子为例，来说明拉纳帕提的方法。

> 美联储主席本·伯南克昨天告诉媒体7 000亿美元的救助资金将借给上百家银行、保险公司和汽车公司。

我们先对它进行分词

> 美联储 | 主席 | 本·伯南克 | 昨天 | 告诉 | 媒体 | 7 000亿 | 美元 | 的 | 救助 | 资金 | 将 | 借给 | 上百 | 家 | 银行 | 、 | 保险公司 | 和 | 汽车公司 |

然后将这些词从左到右扫描一遍，用括号括起来，形成词组：

> （美联储 主席）| 本·伯南克 | 昨天 | 告诉 | 媒体（7 000亿 美元）| 的（救助 资金）（将 借给）（上百 家）（银行 、 保险公司 和 汽车公司）

然后，拉纳帕提给每个括号一个句子成分，比如“美联储主席”是名词短语。接下来，继续重复以上括括号的过程。

> [（美联储 主席） 本·伯南克] 昨天 | 告诉 | 媒体 [（7 000亿 美元） 的（救助 资金）]（将 借给）[（上百 家）（银行 、 保险公司 和 汽车公司）]

直到整个句子都被一个大括号覆盖。每一个括号，就是句子的一个成分，括号之间嵌套关系，就是不同层面的句子成分的构成关系。

拉纳帕提每次从左到右扫描句子的每个词（或者句子成分）时，只需要判断是否属于以下三个操作之一：

> A1. 是否开始一个新的左括号，比如在“美联储”是新括号的开始。
>
> A2. 是否继续留在这个括号中，比如在“保险公司”的位置，是继续留在括号中。
>
> A3. 是否结束一个括号，即标上右括号，比如“资金”的位置是一个括号的结束。

为了判断是采用哪个操作，拉纳帕提建立了一个统计模型 $P(A|\text{prefix})$，其中 A 表示行动，句子前缀 prefix 是指句子从开头到目前为止所有的词和语法成分。最后，拉纳帕提用最大熵模型实现了这个模型。当然，拉纳帕提还用一个统计模型来预测句子成分的种类。

这个方法速度非常快。每次扫描，句子成分的数量就按一定比例减少，因此，扫描的次数是句子长度的对数函数，很容易证明，整个算法和句子长度成正比。从算法复杂度的角度来讲，这个算法的计算时间已经是最优的了。

拉纳帕提的方法讲起来非常简单，但是想到它确实很了不起。（可见好方法在形式上常常是简单的。）可以说，他是真正将句法分析和数学模型联系起来的关键人物。从那个时候起，大部分句法分析的方法都从启发式搜索转到了括括号上。而句法分析的准确性很大程度上取决于统计模型的好坏，以及从 prefix 提取的特征的有效性。前面我们讲过马库斯的博士生柯林斯的博士论文做得非常出色，原因就是他在特征提取上做了深入的研究。

从上个世纪 70 年代出现了统计语言学和统计语言模型，到上个世纪 90 年代拉纳帕提等人通过统计模型将数学和句法分析结合起来，科学家们对于非常“规矩”的语句，比如《华尔街日报》的句子进行句法的分析，正确率可以达到 80% 以上，基本上达到了普通人的水平。2000 年后，随着互联网的普及，读者接触的内容不再是以专业人士严谨的写作为主，很多都是网民们随意创作的内容。这些“写得不是太好”的句子，或者有严重语法错误的句子，以往任何文法分析器，包括柯林斯做的，正确率连 50% 也达不到。

所幸在很多自然语言处理的应用中，并不需要对语句做深入的分析，只要做浅层的分析（Shallow Parsing），比如找出句子中主要的词组以及它们之间的关系即可。于是科学家研究的重点从语句的深层分析转到对语句的浅层分析上。到上世纪 90 年代后，随着计算机计算能力的增强，科

学家们采用了一种新的数学模型工具 —— 条件随机场，大大提高了句子浅层分析的正确率，正确率可达95%，使得句法分析得以应用到很多产品，比如机器翻译上。

2 条件随机场

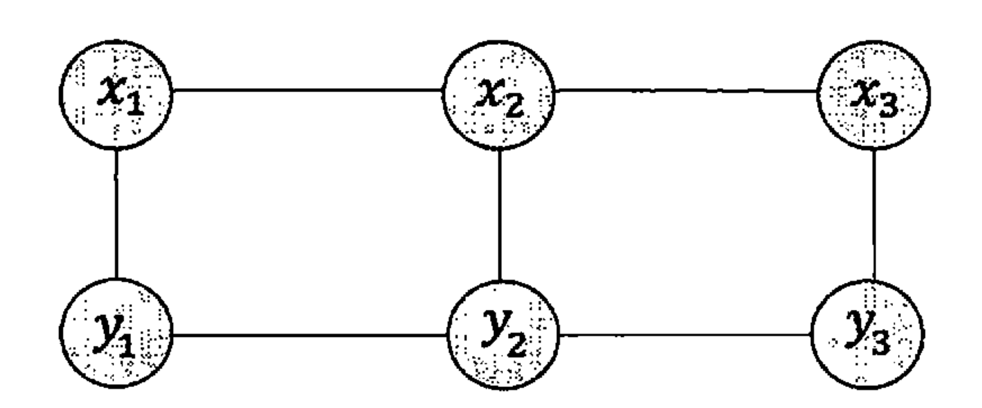


图 25.1 在隐含马尔可夫模型 中，输出只和状态有关

在一个隐含马尔可夫模型中，以 $x_1, x_2, \ldots, x_n$ 表示观测值序列，以 $y_1, y_2, \ldots, y_n$ 表示隐含的状态序列，那么 x_i 只取决于产生它们的状态 y_i，和前后的状态 y_{i-1}，y_{i+1} 都无关。显然在很多应用里观察值 x_i 可能和前后的状态都有关，如果把 x_i 和 y_{i-1}，y_i，y_{i+1} 都考虑进来，对应的模型如下：

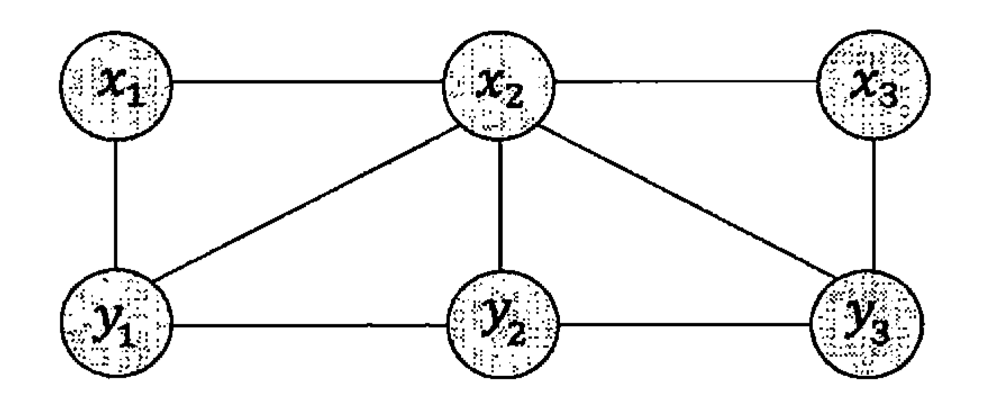


图 25.2 一个普遍意义的条件随机场

这样的模型就是条件随机场。

条件随机场是隐含马尔可夫模型的一种扩展，它保留了隐含马尔可夫模型的一些特性，比如在图中的 $y_1, y_2, \ldots$ 等状态的序列还是一个马尔可夫链。更广义地讲，条件随机场是一种特殊的概率图模型（Probabilistic Graph Model）。在这个图中，顶点代表一个个随机变量，比如 x_1 和 y_1，顶点之间的弧代表它们相互的依赖关系，通常采用一种概率分布，比如 $P(x_1, y_1)$

来描述。它的特殊性在于，变量之间要遵守马尔可夫假设，即每个状态的转移概率只取决于相邻的状态，这一点，它和我们前面介绍的另一种概率图模型 —— 贝叶斯网络相同。而它们的不同之处在于，条件随机场是无向图，而贝叶斯网络是有向图。

在大部分应用中，条件随机场的节点分为状态节点的集合Y和观察变量节点的集合X。整个条件随机场的量化模型就是这两个集合的联合概率分布模型$P(X,Y)$

$$P(X,Y)=P(x_1,x_2,\dots,x_n,y_1,y_2,\dots,y_m) \tag{25.1}$$

由于这个模型的变量特别多，不可能获得足够多的数据来用大数定理直接估计，因此只能通过它的一些边缘分布（Marginal Distribution），比如$P(x_1)$，$P(y_2)$，$P(x_1,y_3)$等等来找出一个符合所有这些条件的概率分布函数。当然，这样的函数可能不止一个（通常如此）。根据最大熵原则，我们希望找到一个符合所有边缘分布，同时使得熵达到最大的模型。前面介绍过，这个模型是指数函数。每一个边缘分布，对应指数模型中的一个特征f_i（Feature），比如针对x_1的边缘分布的特征就是：

$$f_i(x_1,x_2,\dots,x_n,y_1,y_2,\dots,y_m)=f_i(x_1) \tag{25.2}$$

因为这个特征表明它和除x_1以外的变量无关。如果某个特征函数对应一些变量的取值是零，说明这些特征函数对这些变量不起作用。把这些特征都应用到模型中，得到如下的公式：

$$P(x_1,x_2,\dots,x_n,y_1,y_2,\dots,y_m)=\frac{e^{f_1+f_1+\dots+f_k}}{Z} \tag{25.3}$$

以浅层句法分析为例，说明这个条件随机场的模型是如何工作，如何训练的。

假定X代表看到的东西，在浅层分析中，它是句子中的词、每个词的词

性等；Y代表要推导的东西，它是语法成分，比如名词短语、动词短语、时间短语等。以曾经举过的最简单的句子“徐志摩喜欢林徽因”为例来说明浅层分析的模型。这里，观察到的序列是徐志摩／名词，喜欢／动词，林徽因／名词，希望找到的状态序列是“名词短语，动词短语”。我们还是用信道模型来说明这个分析的过程，如图 25.3 所示。

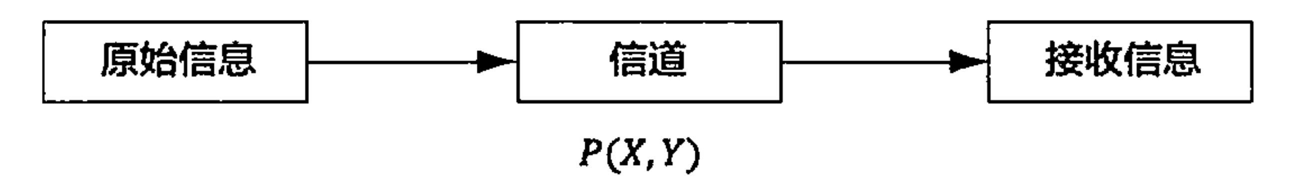


图 25.3　句子文法分析的通信模型

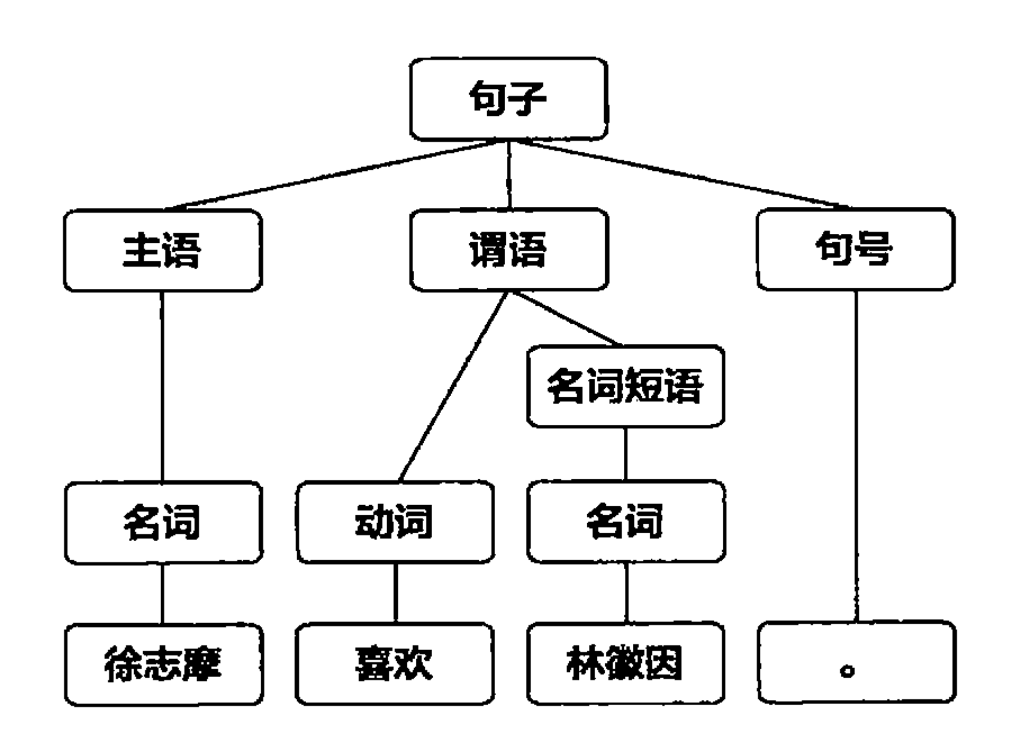


图 25.4　“徐志摩喜欢林徽因”的分析树

在图 25.4 所示的分析树中，它的特征是各个节点、同一层节点顺序的组合、不同层次节点的组合，等等。这里，同一层节点顺序的组合可以是“徐志摩 - 喜欢”、“动词 - 名词”等，不同层次节点的组合其实是一些子树，比如“名词短语”重写（Rewrite）成“名词”，“动词短语”重写成“动词，名词短语”等。在考虑一些特征后，可以用下面的公式计算这个条件随机场的概率

$$
\begin{aligned}
&P(\text{徐志摩/名词，喜欢/动词，林徽因/名词，名词短语，动词短语}) \\
&\quad = \exp\{f_1(\text{徐志摩，名词}) \\
&\quad + f_2(\text{喜欢，动词}) + f_3(\text{林徽因，名词}) \\
&\quad + f_4(\text{徐志摩，名词，名词短语}) + f_5(\text{名词，名词短语}) \\
&\quad + f_6(\text{喜欢，动词，林徽因，名词，动词短语}) \\
&\quad + f_7(\text{名词短语，动词短语}) + f_8(\text{名词短语}) \\
&\quad + f_9(\text{动词短语}) + f_{10}(\text{动词，名词，动词短语})\}
\end{aligned} \tag{25.4}
$$

这里，每一个特征 $f()$ 的参数都可以用前面提到的最大熵模型的训练方法得到。

最早采用条件随机场对句子进行浅层分析的是宾夕法尼亚大学的博士生沙飞（Fei Sha）和他的导师（之一）皮耶尔。他们继承了拉纳帕提的方法，只做了句子分析的第一层，即从词到词组的自动组合。由于改进了统计模型，因此句子浅层分析的正确率高达 94% 左右，这在自然语言处理中是一个很高的水平了。

2005 年，李开复博士来到 Google 后从朱会灿和我的手中接走了中文搜索，我便去领导开发为整个公司服务的通用句子文法分析器 —— Gparser 的项目，其中 G 代表 Google。我们采用的方法跟沙飞和皮耶尔的类似（当时皮耶尔还没有到 Google）。不同的是，我们不仅做第一层的分析，而且可以像拉纳帕提那样，一层层地分析到句子语法树的顶部。基本的模型依然是条件随机场。

在每一层分析中，我们建立了这样一个模型 $P(X,Y)$。其中 X 是句子中的词 $w_1, w_2, \ldots, w_n$、词性 $pos_1, pos_2, \ldots, pos_n$、每一层语法成分的名称 $h_1, h_2, \ldots, h_m$ 等等，Y 是操作（左括号，继续留在括号中，和右括号）以及新的一层语法成分的名称。如果展开写，就是

$$
P\left(w_1, w_2, \ldots, w_n, pos_1, pos_2, \ldots, pos_n, h_1, h_2, \ldots h_m, Y\right) \tag{25.5}
$$

这个看似包罗万象的模型，在工程上有很大的问题。首先，这是一个非

常复杂的模型，因此它的训练是个大问题。其次，各种条件的组合数是天文数字，即使有 Google 的数据，其中任何一个组合可能都出现不了几次。因此需要解决下面两个问题。

首先，做一个近似，把限制条件的组合$w_1, w_2, \ldots, w_n, pos_1, pos_2, \ldots, pos_n, h_1, h_2, \ldots, h_m$拆成很多子集，比如最后的两个词$w_{n-1}, w_n$，最后两个句子成分$h_{m-1}, h_m$等等，在每一个子集和要预测的操作（以及更高层次的语法成份名称）之间可以找到可靠的统计关系。

其次，在训练数据中把每一个统计量足够的统计关系作为一个限制条件，我们的目标是寻找符合所有这些限制条件的最大熵模型。

这样，就可以用最大熵模型的方法，得到这个语法分析器了。

这样得到的语法分析器，对于网页数据，句子浅层分析的结果接近沙飞论文对《华尔街日报》的准确性，但是可以处理很“脏”的网页数据。这个分析器，先后被用到 Google 的许多产品中。

3 小结

条件随机场是一个非常灵活的用于预测的统计模型。虽然在本章中我们强调了它在自然语言处理，特别是在句子分析中的应用，但是它的应用领域远远不止这些。它在模式识别、机器学习和生物统计中都有很成功的应用。

和最大熵模型一样，条件随机场的形式简单，但是实现复杂，所幸的是现在有不少开源的软件可供使用，对于一般的工程人员应该是足够了。

第26章 维特比和他的维特比算法

说起安德鲁·维特比（Andrew Viterbi），不是通信行业的人可能知道他的并不多，虽然很多通信行业的从业者大多知道以他的名字命名的维特比算法（Viterbi Algorithm）。维特比算法是现代数字通信中使用最频繁的算法，同时也是很多自然语言处理的解码算法。可以毫不夸张地讲，维特比是对我们今天的生活影响力最大的科学家之一，因为今天基于 CDMA 的 3G 移动通信标准主要就是他和欧文·雅克布斯（Irwin Mark Jacobs）创办的高通公司（Qualcomm）制定的。

1 维特比算法

我第一次听到维特比这个名字还是 1986 年学习图论的维特比算法时了解到的，那一年他正在和雅克布斯博士刚刚创立起他们的第二家公司——高通公司（1985 年注册）。截至那时，世界对他的了解还仅仅在学术界，具体来说是通信界和计算机算法领域。我第一次使用这个算法是 1991 年从事语音识别研究的时候，那一年高通公司已经提出和完善了今天 3G 通信的基础——CDMA 的各项协议。而 1997 年夏天，我在约翰·霍普金斯大学第一次见到维特比博士时，他已经是名满天下的通信工业界巨子了。那年他作为 CLSP 的顾问来参加年会，听我们介绍在 CLSP 的工作。他最关心的是语音识别的商业前景，今天他可以看到了。

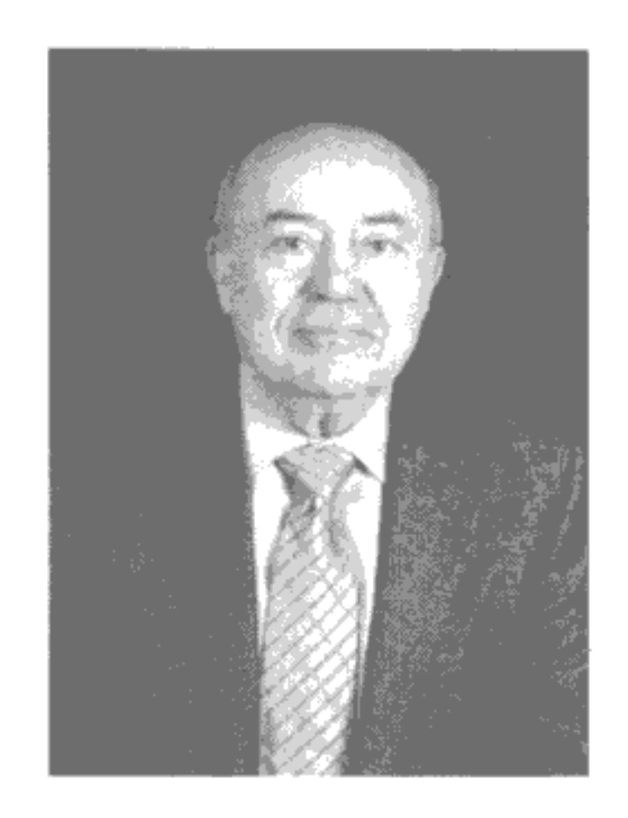


图 26.1 科学家和商业巨子维特比

维特比博士是美籍意大利犹太移民，他原名叫 Andrea Viterbi，但是 Andrea 这个名字在英语里是个女孩子的名字，因此他把自己的名字改成安德鲁（Andrew）。从他在麻省理工学院毕业到他 33 岁以前，维特比的职业生涯完全局限在学术界。他先后在著名的国防工业公司雷神（Raytheon）、美国著名的喷气推进实验室（JPL）担任工程师，同时在南加州大学（University of Southern California）完成了博士学位。之后在加州大学（洛杉矶和圣地亚哥两个分校）担任教职，从事刚刚兴起的学科 —— 数字通信的研究，几年后的 1967 年，他发明了维特比算法。

言归正传，现在让我们看看作为科学家的维特比得以成名的算法。维特比算法是一个特殊但应用最广的动态规划算法（见前面相应的章节）。利用动态规划，可以解决任何一个图中的最短路径问题。而维特比算法是针对一个特殊的图 —— 篱笆网络的有向图（Lattice）的最短路径问题而提出的。它之所以重要，是因为凡是使用隐含马尔可夫模型描述的问题都可以用它来解码，包括今天的数字通信、语音识别、机器翻译、拼音转汉字、分词等。下面还是拿大家用得最多的输入法拼音转汉字来说明。

假定用户（盲打时）输入的拼音是 $y_1, y_2, \ldots, y_N$，对应的汉字是 $x_1, x_2, \ldots, x_N$（虽然真正的输入法产品都是以词作为输入单位的，但是为了便于说明问题简单起见，以字为单位来解释维特比算法），那么根据前面介绍的工具：

$$x_1,x_2,\dots,x_N = \underset{x\in X}{\text{ArgMax}}\, P(x_1,x_2,\dots,x_N|y_1,y_2,\dots,y_N)$$
$$= \underset{x\in X}{\text{ArgMax}} \prod_{i=1}^{N} P(y_i|x_i)\cdot P(x_i|x_{i-1}) \qquad (26.1)$$

输入的（可见）序列为 $y_1,y_2,\dots,y_N$，而产生它们的隐含序列是 $x_1,x_2,\dots,x_N$。可以用下图描述这样一个过程：

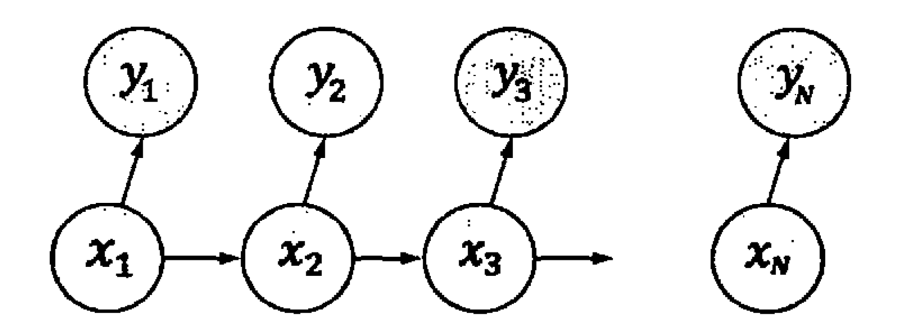


图 26.2　适用维特比算法的隐含马尔可夫模型

这是一个没有状态跳跃，也没有状态自环的相对简单的隐含马尔可夫链。$P(x_i|x_{i-1})$ 是状态之间的转移概率，$P(y_i|x_i)$ 是每个状态的产生概率。现在，这个马尔可夫链的每个状态的输出是固定的，但是每个状态的值可以变化。比如输出读音“zhong”的字可以是“中”、“种”等多个字。我们不妨抽象一点，用符号 x_{ij} 表示状态 x_i 的第 j 个可能的值。如果把每个状态按照不同的值展开，就得到下面这个篱笆网络（Lattice）：

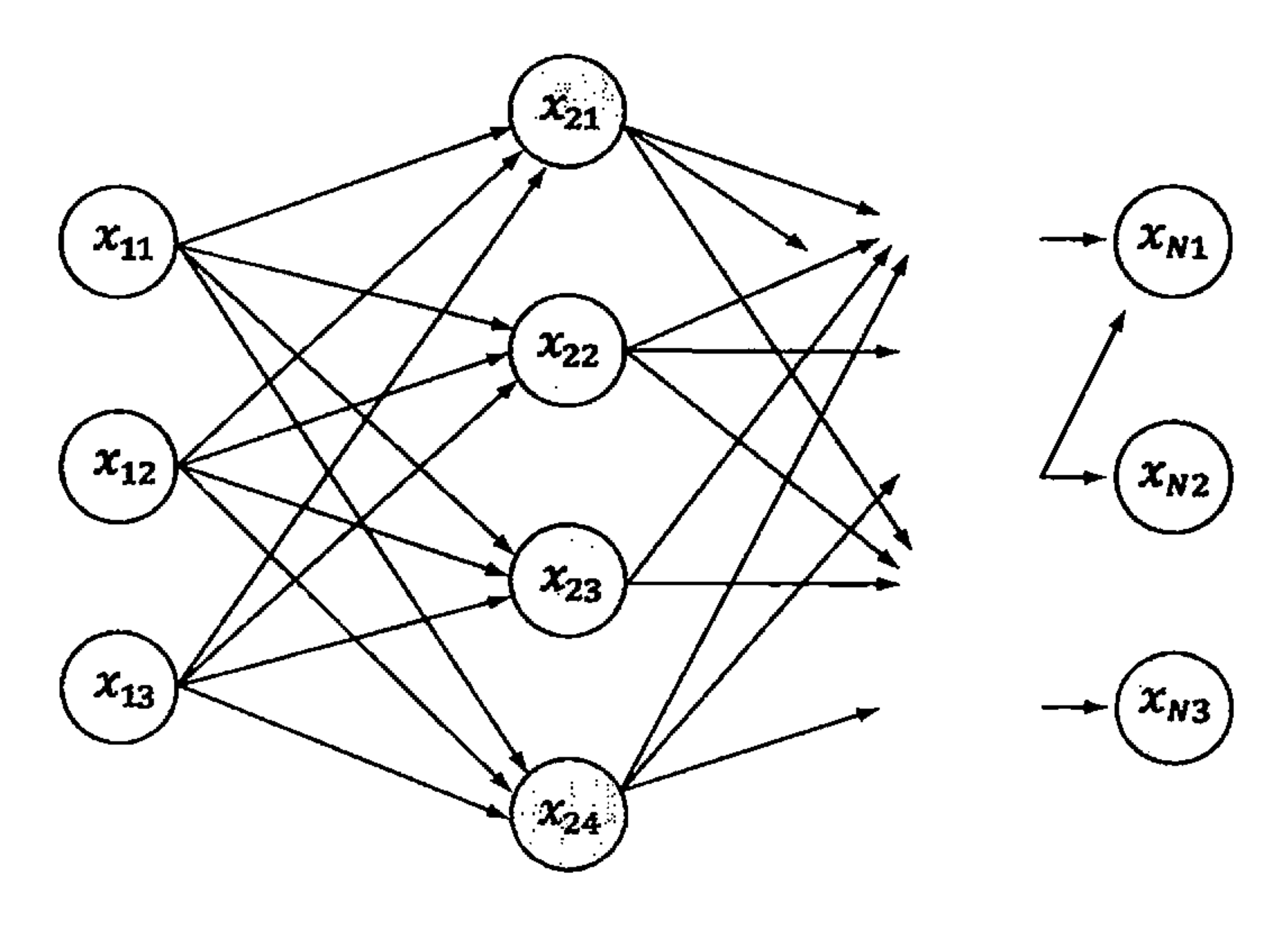


图 26.3　篱笆网络

在上图中，每个状态有 3 个或 4 个值，当然实际中它们可以有任意个值。

那么从第一个状态到最后一个状态的任何一条路径（path）都可能产生我们观察到的输出序列Y。当然，这些路径的可能性不一样，而我们要做的就是找到最可能的这条路径。对于每一条给定的路径，可以使用公式（26.1）计算出它的可能性，这不是件困难的事情。但麻烦的是这样的路径组合数非常多，它使得序列状态数的增长呈指数爆炸式。汉语中每个无声调的拼音对应 13 个左右的国标汉字，假定句长为 10 个字，那么这个组合数为 $13^{10}\sim5\times10^{14}$。假定计算每条路径概率需要 20 次乘法（或者加法，如果程序设计者足够聪明的话），就是 10^{16} 次计算。而今天的微机处理器每秒大约能计算 10^{11} 次，也需要大约一天时间。而在通信或语音识别中，每说一句话的状态数是成千上万，上面的穷举法显然不合适。因此，需要一个最好能和状态数目成正比的算法。而这个算法是维特比在 1967 年首次提出的，因此就被命名为维特比算法。

维特比算法的基础可以概括成下面三点：

1. 如果概率最大的路径P（或者说最短路径）经过某个点，比如图中的x_{22}，那么这条路径上从起始点S到x_{22}的这一段子路径Q，一定是S到x_{22}之间的最短路径。否则，用S到x_{22}的最短路径R替代Q，便构成了一条比P更短的路径，这显然是矛盾的。

2. 从S到E的路径必定经过第i时刻的某个状态（这显然是大白话，但是很关键），假定第i时刻有k个状态，那么如果记录了从S到第i个状态的所有k个节点的最短路径，最终的最短路径必经过其中的一条。这样，在任何时刻，只要考虑非常有限条最短路径即可。

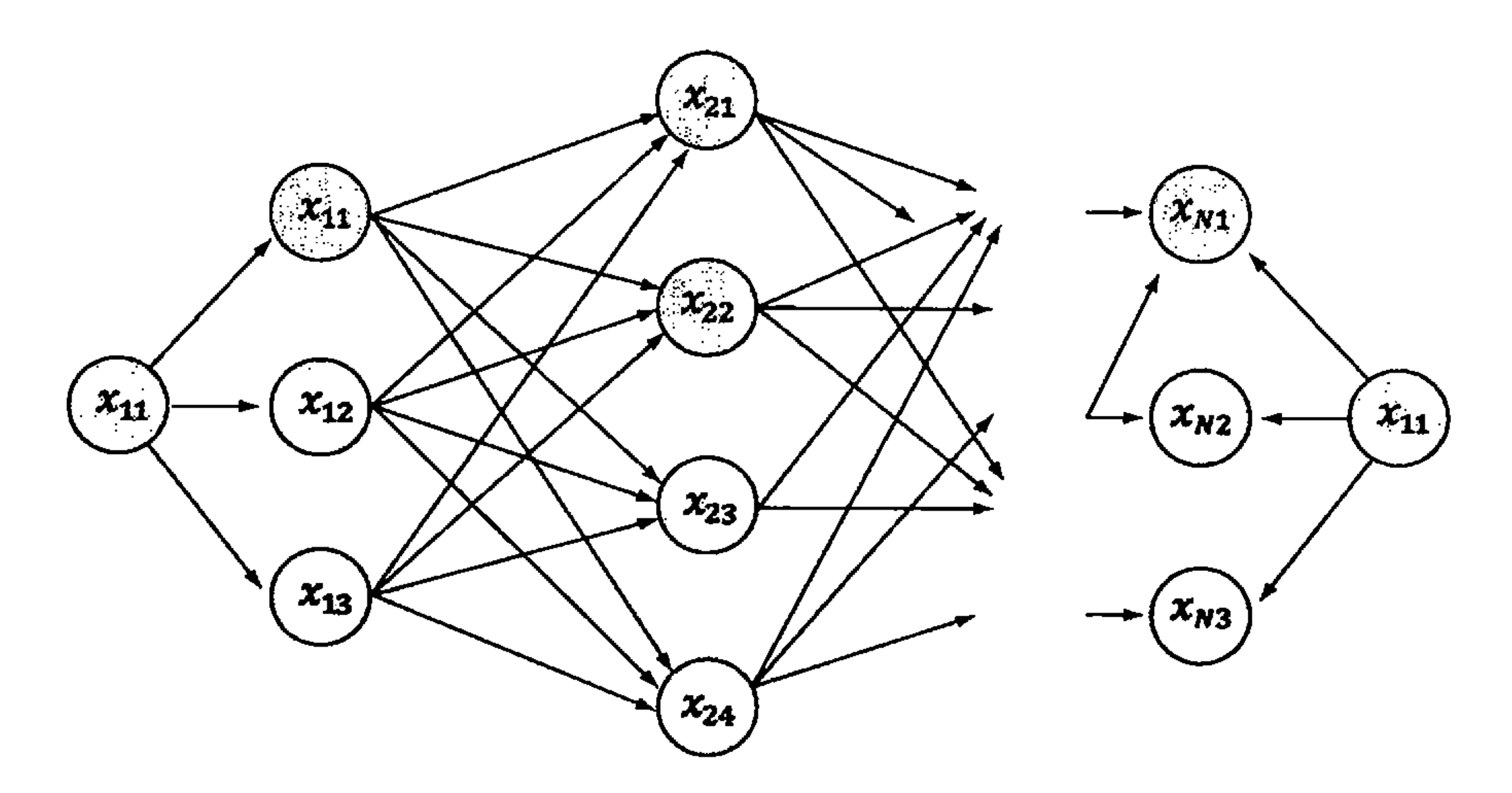


图 26.4　从起点到终点的路径必定经过第 i 时刻的某个状态

3. 结合上述两点，假定当我们从状态 i 进入状态 $i+1$ 时，从 S 到状态 i 上各个节点的最短路径已经找到，并且记录在这些节点上，那么在计算从起点 S 到第 $i+1$ 状态的某个节点 x_{i+1} 的最短路径时，只要考虑从 S 到前一个状态 i 所有的 k 个节点的最短路径，以及从这 k 个节点到 x_{i+1}，j 的距离即可。

基于上述三点基础，维特比总结了如下的算法：

第一步，从点 S 出发，对于第一个状态 x_1 的各个节点，不妨假定有 n_1 个，计算出 S 到它们的距离 $d(S,x_{1i})$，其中 x_{1i} 代表任意状态 1 的节点。因为只有一步，所以这些距离都是 S 到它们各自的最短距离。

第二步，这是理解整个算法的关键。对于第二个状态 x_2 的所有节点，要计算出从 S 到它们的最短距离。我们知道，对于特定的节点 x_{2i}，从 S 到它的路径可以经过状态 1 的 n_1 中任何一个节点 x_{1i}，当然，对应的路径长度就是 $d(S,x_{2i}) = d(S,x_{1j}) + d(x_{1j},x_{2i})$。由于 j 有 n_1 种可能性，我们要一一计算，然后找到最小值。即

$$d(S,x_{2i}) = \min_{I=1,n_1} d(S,x_{1j}) + d(x_{1j},x_{2i}) \qquad (26.2)$$

这样对于第二个状态的每个节点，需要n_1次乘法计算。假定这个状态有n_2个节点，把S这些节点的距离都算一遍，就有$O(n_1 \cdot n_2)$次计算。

接下来，类似地按照上述方法从第二个状态走到第三个状态，一直走到最后一个状态，就得到了整个网格从头到尾的最短路径。每一步计算的复杂度都和相邻两个状态S_i和S_{i+1}各自的节点数目n_i，n_{i+1}的乘积成正比，即$O(n_i \cdot n_{i+1})$。如果假定在这个隐含马尔可夫链中节点最多的状态有D个节点，也就是说整个网格的宽度为D，那么任何一步的复杂度不超过$O(D^2)$，由于网格长度是N，所以整个维特比算法的复杂度是$O(N \cdot D^2)$。

回到上面那个输入法的问题，计算量基本上是 $13 \times 13 \times 10 = 1690 \approx 10^3$，这和原来的$10^{16}$有天壤之别。更重要的是，维特比算法是和长度成正比的。无论是在通信中，还是在语音识别、打字中，输入都是按照流（Stream）的方式进行的，只要在处理每个状态的时间比讲话或者打字速度快（这点很容易做到），那么不论输入有多长，解码过程永远是实时的。这便是数学漂亮的地方！

虽然数字通信和自然语言处理的基础算法的原理其实就是这么简单，每个通信或者计算机专业的学生两个小时就能学会，但是在上个世纪 60 年代能够想出这个快速算法是非常了不起的。凭借这个算法，维特比奠定了他在数字通信中不可替代的地位。但是，维特比并不满足停留在算法本身，而是努力将它推广出去。维特比为此做了两件事：首先，他放弃了这个算法的专利；第二，他和雅克布斯博士一起在 1968 年创办了 Linkabit 公司，将这个算法做成芯片，卖给其他通信公司。

到这一步维特比已经比一般的科学家走得远很多了。但是，这仅仅是维特比辉煌人生的第一步。上个世纪 80 年代，维特比致力于将 CDMA 技术应用于无线通信。

2 CDMA 技术 — 3G 移动通信的基础

自从苹果推出 iPhone，3G 手机和移动互联网就成为科技界和工业界的热点话题。这里面最关键的通信技术就是码分多址（CDMA）技术。对 CDMA 技术的发明和普及贡献最大的有两个人 — 奥匈帝国出生、美籍犹太裔的海蒂·拉玛尔（Hedy Lamarr）和维特比。



图 26.5 海蒂·拉玛尔

拉玛尔被誉为史上最美丽的科学家，其实她的主要职业是演员，通信上调频技术的发明是她的副业。拉玛尔从小（10 岁）学习舞蹈和钢琴，并因此进入了演艺界。拉玛尔在演奏钢琴时，想到用钢琴不同键所发出的不同频率来对信号进行加密。如果接收者知道调频的序列就可以解码收到的信号，如果不知道这个序列，就无法破解。这就像如果你听过并记得肖邦的"英雄波兰舞曲"，你就知道演奏的是什么，否则就是一些凌乱的音符而已。拉玛尔和她的邻居、作曲家乔治·安泰尔（George Antheil）一道发明了一种称为"保密通信系统"的调频通信技术。在这种技术中，通信信号的载波频率是快速跳变的，只要发送方和接收方事先约定一个序列（一般是一个伪随机数序列）即可。想截获信息的人因为不知道这个序列而无能为力。拉玛尔最早是采用钢琴的 88 个键的频率做载波频率，将约定好的调频序列做在钢琴卷（Piano Roll）[1] 上，然后载波频率根据钢琴卷上的打孔位置而变化。

1 钢琴卷是一个自动控制钢琴演奏的纸卷，像早期计算机使用的纸带。上面打了眼，表示不同的音符，读卷机可以因此知道音符，并控制钢琴。这有点像今天的 MIDI 的钢琴自动演奏器。

这种调频技术是今天的CDMA的前身，它于1941年获得美国专利。美国海军曾经想用这项技术实现一个敌人无法发现的无线电控制的鱼雷，但是因为有反对意见暂时搁起。很快二战就结束了，直到1962年都没有实现这项技术。越战期间，越南军方发现被击落的美国飞行员可以通过一种检查不出频率的设备呼救。他们缴获这种设备后，搞不清它的原理，也不知道如何能破获它产生的信号，于是他们把这个设备交给援越的中国顾问团。中国顾问团里有一些通信专家，包括我在清华大学的导师王作英教授，他们发现这种设备能以极低的功率在很宽的频带上发送加密信号。对于试图截获者来讲，这些信号能量非常低，很难获取。即使能够获得，因为不知道密码而无法破解。但是对于接收者来讲，它可以通过把很低的能量积累起来获得发送的信息，并且因为知道密钥，实现解码。

这种传输方式是在一个较宽的扩展频带上进行的，因此它称为扩频传输（Spread-Spectrum Transmission）。和固定频率的传输相比，它有三点明显的好处。

1. 它的抗干扰能力极强。生活在文革前后年代的人可能知道，过去禁止收听外国的广播。这条规矩很难实施，因为无线电波满天都是，很容易收听。因此，政府可以做的事就是用噪音干扰那些固定的广播频率。但是对于扩频传输，这件事几乎不可能，因为不能把所有的频带都干扰了，这样整个国家的通信就中断了。

2. 扩频传输的信号很难被截获，这点前面已经讲了。

3. 扩频传输利用带宽更充分。这一点详细解释起来有点啰嗦，简单地讲就是固定频率的通信由于邻近的频率互相干扰，载波频率的频点不能分布得太密集，两个频点之间的频带就浪费了。扩频通信由于抗干扰能力强，浪费的频带较少。

虽然扩频技术和调频技术早在上个世纪60年代就应用于军事，但是转为民用则是上个世纪80年代以后的事情。这首先要归功于移动通信的需求。上个世纪80年代，移动通信开始快速发展，很快大家发现空中的频带不够用了，需要采用新的通信技术。其次，具体到CDMA本身，很大程度上归功于维特比的贡献。

在 CDMA 以前，移动通信使用过两种技术：频分多址（FDMA）和时分多址（TDMA）技术。

频分多址顾名思义，是对频率进行切分，每一路通信使用一个不同的频率，对讲机采用的就是这个原理。由于相邻频率会互相干扰，因此每个信道要有足够的带宽。如果用户数量增加，总带宽就必须增加。我们知道空中的频带资源是有限的，因此要么必须限制通信人数，要么降低话音质量。

时分多址是将同一频带按时间分成很多份。每个人的（语音）通信数据在压缩后只占用这个频带传输的 1 / N 时间，这样同一个频带可以被多个人同时使用。第二代移动通信的标准都是基于 TDMA 的。

前面讲了，扩频传输对频带的利用率比固定频率传输更高，因此，如果把很多细分的频带合在一起，很多路信息同时传输，那么应该可以提高带宽的利用率，这样就可以增加用户的数量，或者当用户数量不变时，提高每个用户的传输速度。

美国的两个主要无线运营商 AT&T 和 Verizon，前者的基站密度不比后者低，信号强度也不比后者差，但是通话质量和数据传输速度却明显不如后者，原因就是 AT&T 网络总的来讲是继承过去 TDMA 的，而 Verizon 则完全是基于 CDMA 的。

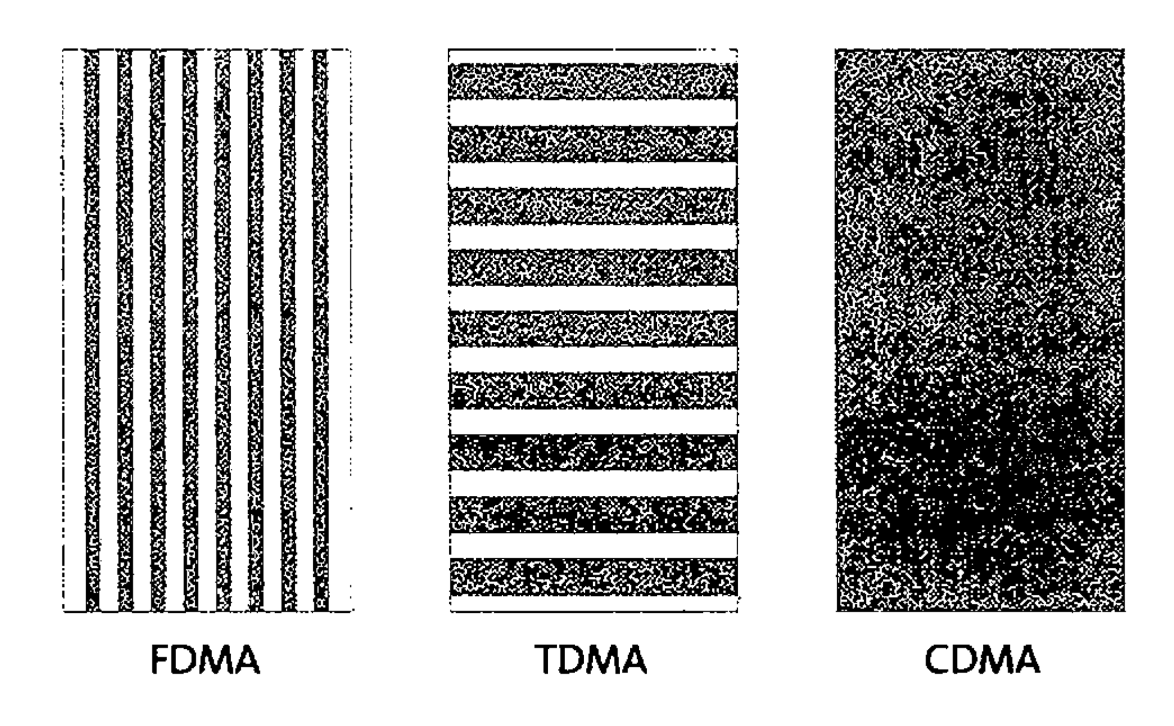


图 26.6　频分多址（FDMA）、时分多址（TDMA）和码分多址（CDMA）对频带和时间的利用率，图中深色的部分有可利用部分，边界无色的部分为不可利用部分。

当然，读者可能会有个问题：如果一个发送者占用了很多频带，那么有多个发送者同时发射岂不打架了？没关系，每个发送者有自己不同的密码，接收者在接到不同信号时，通过密码过滤掉自己无法解码的信号，留下和自己密码对应的信号即可。由于这种方法是根据不同的密码区分发送的，因此称为码分多址。

将 CDMA 技术用于移动通信的是高通公司。从 1985 年到 1995 年，高通公司制定和完善了 CDMA 的通信标准 CDMA1，并于 2000 年发布了世界上第一个主导行业的 3G 通信标准 CDMA2000，后来又和欧洲、日本的通信公司一同制定了世界上第二个 3G 标准 WCDMA。2007 年，维特比作为数学家和计算机科学家，被授予美国科技界最高成就奖 —— 国家科学奖[2]。

2 http://www.nsf.gov/od/nms/recip_details.cfm?recip_id=5300000000446



图 26.7　小布什总统授予维特比博士国家科学奖

或许是因为维特比极强的技术背景，高通公司完全是纯技术基因。虽然高通公司是今天世界上最大的 3G 手机处理器厂商，但是它没有制造，只有研发和设计。而它的很大一部分利润来自于专利费。或许是由于过分强调技术，它在第二代移动通信的竞争中，输给了欧洲的公司，因为那时快速数据传输对移动用户来讲不是最必需的需求。但是，高通技术上的优势保证了它对第三代移动通信的统治地位。

如果把维特比算作数学家中的一员，那么他也许是全世界有史以来第二

富有的数学家（第一富有的无疑是文艺复兴技术公司的创始人吉姆·赛蒙斯）。维特比是南加州大学最大的资助者之一[3]，该校的工学院也是以他的名字命名的。他的财富来自于他将技术转换成商业的成功。

3 维特比向南加州大学捐赠了 5200 万美元。



图 26.8　南加州大学的维特比工学院

3　小结

世界上绝大多数科学家最大的满足就是自己的研究成果得到同行的认可，如果能有应用就更是喜出望外了。而能够亲自将这些成就应用到实际中的人少之又少，因为做到这一点对科学家来讲很不容易。这样的科学家包括 RISC 的发明人亨利西和 DSL 之父查菲等人。这些人已经非常了不起了，但是也只做了一个行业中他们擅长的部分，而不是从头到尾完成一次革命。而维特比所做的远远超出了这一点，他不仅提供了关键性的发明，而且为了保障这项关键性的发明的效益在全社会得到最大化，他解决了所有配套的技术。所有试图另辟蹊径的公司都发现，高通公司的标准几乎无法绕过去，因为他们已经把能想到的事情都想到了。

第27章　再谈文本自动分类问题——期望最大化算法

前面的章节已经多次讨论到文本自动分类问题，一方面是因为今天互联网的各种产品和应用都需要用到这个技术，另一方面，这个技术可以用到几乎所有分类中，比如用户的分类、词的分类、商品的分类，甚至生物特征和基因的分类，等等。这一章再介绍一些文本自动分类的技术，并借此说明在机器学习中最重要的一个方法——期望最大化算法（Expectation Maximization Algorithm）。这个算法，我称之为上帝的算法。

1　文本的自收敛分类

关于文本 TF-IDF 向量的计算和余弦距离的计算就不再重复了，在分类中依然用这个特征来度量。和前面章节介绍的方法不同的是，这里既不需要事先设定好类别，也不需要对文本两两比较进行合并聚类。而是随机地挑出一些类的中心（Centroids），然后来优化这些中心，使它们和真实的聚类中心尽可能一致。

这种自收敛的分类说起来很简单。假设有N篇文本，对应N个向量$V_1, V_2, \cdots, V_N$，希望把它们分到K类中，而这K类的中心是$c_1, c_2, \ldots, c_K$。无论是这些向量，还是中心，都可以看成是空间中的点。当然K可以是一个固定的数，比如我们认为文本的主题只有 100 类；也可以是一个不定

的数，比如我们并不知道文本的主题有多少类，最终有多少类就分成多少类。分类的步骤如下：

1. 随机挑选K个点，作为起始的中心$c_1(0), c_2(0), \ldots, c_K(0)$，如图27.1中各个点属于三个类，我们用黑十字代表随机指定的类的中心。

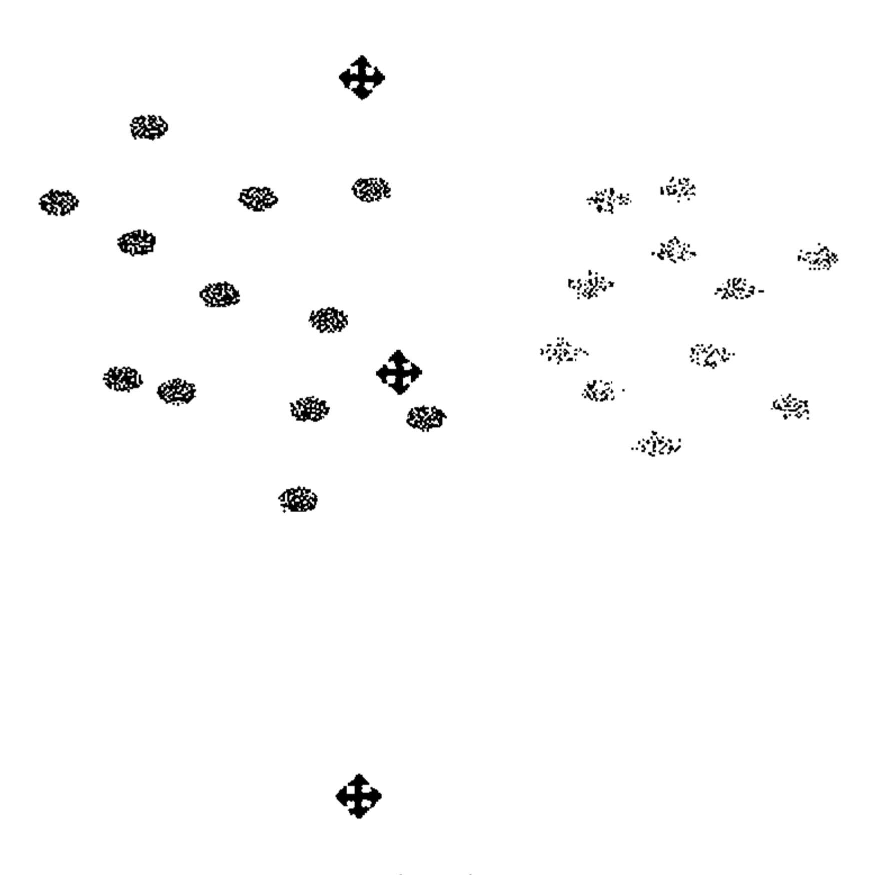


图 27.1 三个类的自动分类

2. 计算所有点到这些聚类中心的距离，将这些点归到最近的一类中。如下图：

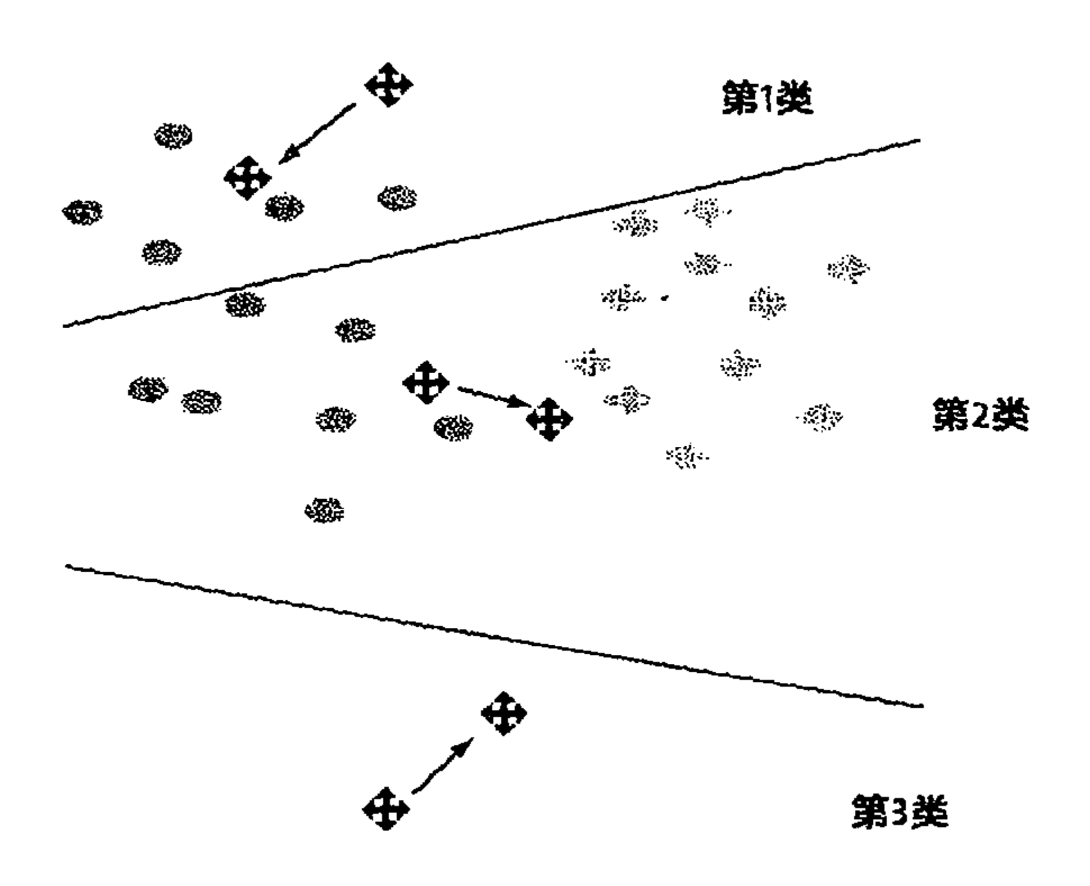


图 27.2 根据中心对每个点重新分类，并计算新的中心

3. 重新计算每一类的中心。假定某一类中的 v，每一个点有多个维度，即

$$v_1 = v_{11}, v_{12}, \ldots, v_{1d}$$
$$v_2 = v_{21}, v_{22}, \ldots, v_{2d}$$
$$\ldots$$
$$v_M = v_{m1}, v_{m2}, \ldots, v_{md}$$

最简单的办法就是用这些类的中心 $w = w_1, w_2, \ldots, w_m$ 作为其中心，其中第 i 维的值计算如下：

$$w_i = \frac{v_{1i}+v_{2i}+\cdots+v_{mi}}{m} \tag{27.1}$$

新的聚类中心和原先的相比会有一个位移，图 27.2 中用箭头表示了中心的移动。箭头指向处为新的聚类中心。

4. 重复上述过程，直到每次新的中心和旧的中心之间偏移非常非常小，即过程收敛。

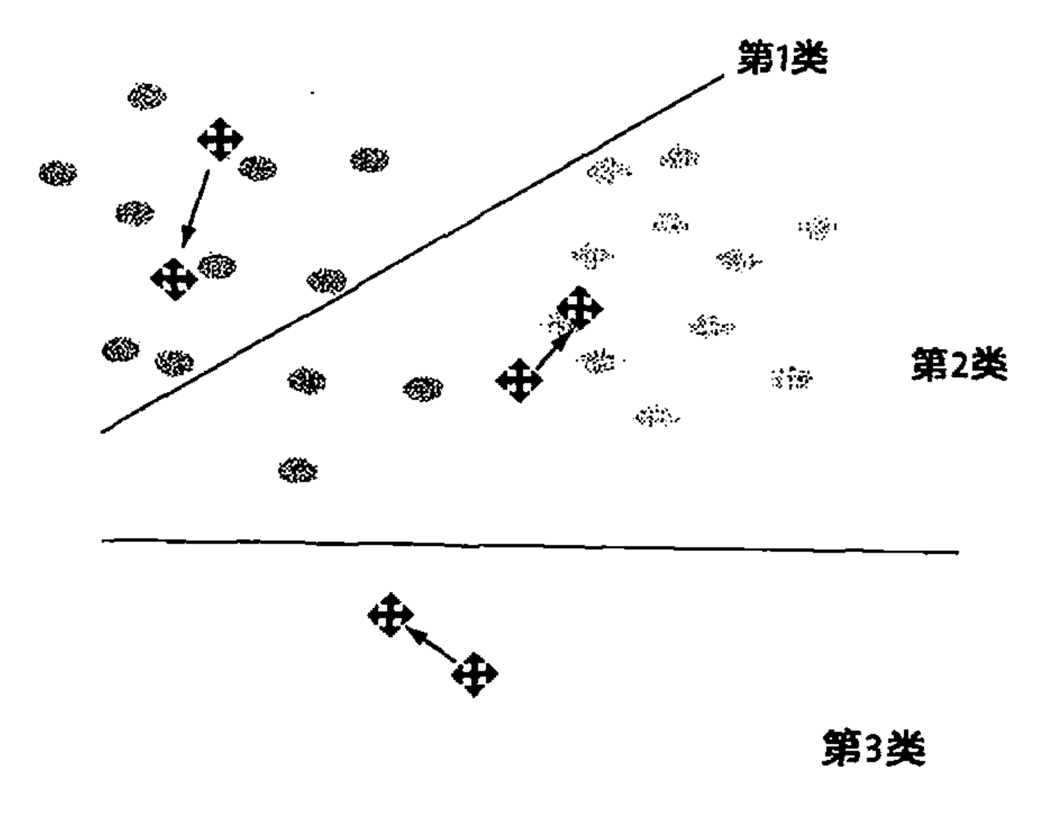


图 27.3　第二次迭代后的结果

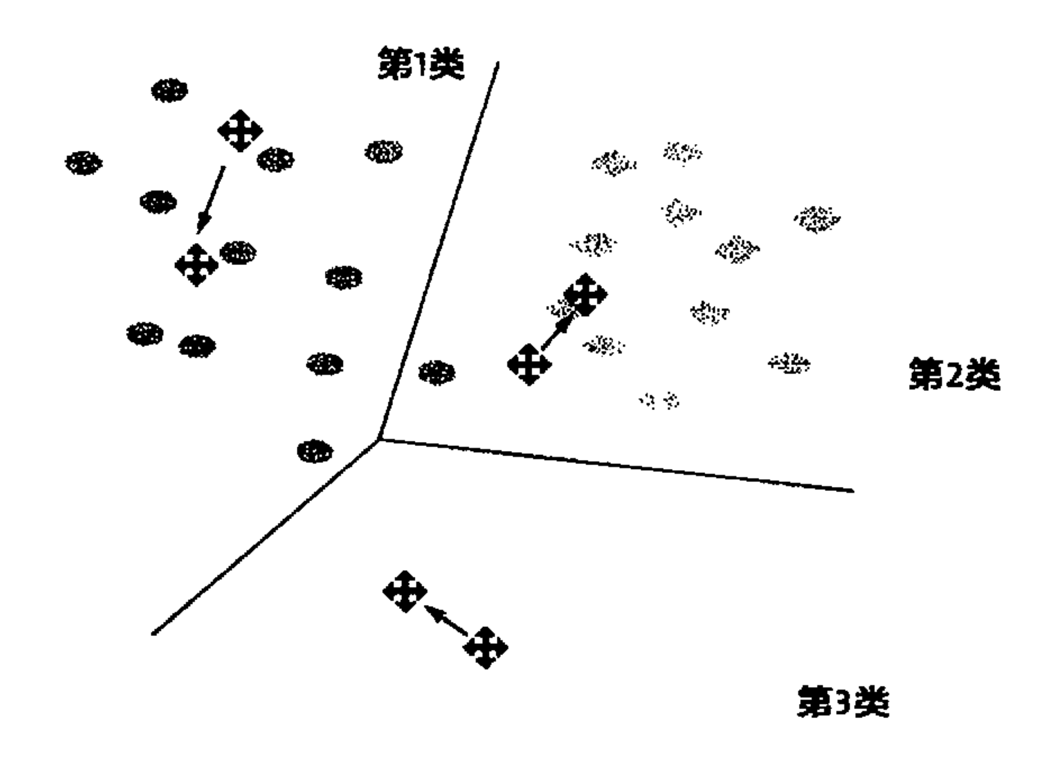


图 27.4 第三次迭代后，基本已经收敛

上面这个例子是我随机产生的，大家从这几张图可以看出，聚类的过程经过几次迭代就可以收敛了。这个方法不需要任何人工干预和先验的经验（Prior Experience），是一些纯粹的数学计算，最后就完全得到了自动的分类。这简直有点令人难以置信，读者可能会问这样做是否就能保证将距离近的点聚集在一起？如果能，为什么？

2 延伸阅读：期望最大化和收敛的必然性

读者背景知识：机器学习或者模式分类。

首先要明确一点，就是我们的距离函数足够好，它能保证同一类相对距离较近，而不同类的相对距离较远。我们希望最终的分类结果是：相近的点都被聚集到了一类中，这样同一类中各个点到中心的平均距离 d 较近，而不同类中心之间的平均距离 D 较远。我们希望的迭代过程是每一次迭代时，d 比以前变小，而 D 变大。

假定第 1 类到第 K 类中分别有 $n_1, n_2, \dots, n_k$ 个点，每一类中，点到中心的平均距离是 $d_1, d_2, \dots, d_k$，因此 $d = (n_1 \cdot d_1 + n_2 \cdot d_2 + \cdots + n_k \cdot d_k)/k$ 。继续假定第 i 类和第 j 类中心之间的距离是 D_{ij}，那么 $D = \sum_{i,j} \frac{D_{ij}}{k(k-1)}$。如果考虑到不同类的大小，即点的数量，那么 D 加权平均的公式应该是

$$D = \sum_{i,j} \frac{D_{ij} n_i n_j}{n(n-1)} \tag{27.2}$$

假定有一个点 x，它在前一次迭代中属于第 i 类，但是在下一次迭代中，它和第 j 类距离更近，根据我们的算法，它将被安排到第 j 类中。不难证明 $d(i+1) < d(i)$，同时 $D(i+1) > D(i)$。

好了，知道了每一步迭代后，我们都离目标（最佳分类）更近了一步，直到最终达到最佳分类。

可以把上面的思想扩展到更一般的机器学习问题中。上述算法实际上包含两个过程和一组目标函数。这两个过程是：

1. 根据现有的聚类结果，对所有数据（点）重新进行划分。如果把最终得到的分类结果看作是一个数学的模型，那么这些聚类的中心（值），以及每一个点和聚类的隶属关系，可以看成是这个模型的参数。
2. 根据重新划分的结果，得到新的聚类。

而目标函数就是上面的点到聚类的距离 d 和聚类之间的距离 D，整个过程就是要最大化目标函数。

在一般性的问题中，如果有非常多的观测数据（点），类似上面的方法，让计算机不断迭代来学习一个模型。首先，根据现有的模型，计算各个观测数据输入到模型中的计算结果，这个过程称为期望值计算过程（Expectation），或 E 过程；接下来，重新计算模型参数，以最大化期望值。在上面的例子中，我们最大化 D 和 $-d$，这个过程称为最大化的过程（Maximization），或 M 过程。这一类算法都称为 EM 算法。

前面介绍过的很多算法，其实都是 EM 算法。比如隐含马尔可夫模型的训练方法 Baum-Welch 算法，以及最大熵模型的训练方法 GIS 算法。在 Baum-Welch 算法中，E 过程就是根据现有的模型计算每个状态之间转

移的次数（可以是分数值）以及每个状态产生它们输出的次数，M 过程就是根据这些次数重新估计隐含马尔可夫模型的参数。这里最大化的目标函数就是观测值的概率。在最大熵模型的通用迭代算法 GIS 中，E 过程就是跟着现有的模型计算每一个特征的数学期望值，M 过程就是根据这些特征的数学期望值和实际观测值的比值，调整模型参数。这里，最大化的目标函数是熵函数。

最后还要讨论一点，就是 EM 算法是否一定能保证获得全局最优解？如果我们优化的目标函数是一个凸函数，那么一定能保证得到全局最优解。所幸的是我们的熵函数是一个凸函数，如果在 N 维空间以欧氏距离做度量，聚类中我们试图优化的两个函数也是凸函数。但是，对应的很多情况，包括文本分类中的余弦距离都不保证是凸函数，因此有可能 EM 算法给出的是局部最佳解而非全局最佳解。



图 27.5 爬到了山顶，那个山头好像更高，可是我已经找不到向上的路了

3 小结

EM 算法只需要有一些训练数据，定义一个最大化函数，剩下的事情就交给计算机了。经过若干次迭代，我们需要的模型就训练好了。这实在是太美妙了，这也许是造物主刻意安排的。所以我把它称作上帝的算法。

第28章　逻辑回归和搜索广告

搜索广告之所以比传统的在线展示广告（Display Ads）赚钱多很多，除了搜索者的意图明确外，更重要的是靠预测用户可能会点击哪些广告，来决定在搜索结果页中插入哪些广告。

1　搜索广告的发展

搜索广告基本上走过了三个阶段。第一个阶段是以早期 Overture 和百度的广告系统为代表，按广告主出价高低来排名的竞价排名广告。简单地说，就是谁给钱多，就优先展示谁的广告。为了支持这种做法，雅虎还给出了一个假设，出得起价钱的公司一定是好公司，因此不会伤害用户体验。但是这个假设等同于好货一定能淘汰劣货，事实并非如此，出得起价钱的公司常常是卖假药的公司，因为它们获利最丰。这样一来就破坏了用户体验，因此很快用户就不点这些广告了，久而久之，所有的广告用户都不点了。如果这样持续多年，没有了点击量，广告商也就不来了。这个行业就要萎缩。

事实上，这种看上去挣钱多的方法，并没有给雅虎带来比 Google 更多的利润。相反，它的单位搜索量带来的收入（一般以千次搜索量带来的收入来衡量，称为 RPM）不到 Google 的一半。而 Google 反而不一定将出

价高的广告放在前面，而是预测到哪个广告可能被点击，结合出价和点击率(Click Through Rate，简称 CTR)这两点来决定广告的投放。几年后，雅虎和百度看到自身和 Google 的差距，也学着 Google 的方法做，就是所谓的“Panama 系统”和“凤巢系统”。它们相当于 Google 的第一个版本，可以看作是搜索广告的第二个阶段。这里面的关键技术就是预测用户可能点击候选广告的概率，或者称为点击率预估。第三个阶段其实是一个全局的进一步优化，和这一章的主题无关，就不赘述了。

点击率预估最好的办法，就是根据以前的经验值来预测。比如对特定的查询，广告 A 展示了 1000 次，被点击 18 次，广告 B 展示 1200 次，被点击 30 次，那么它们的点击率分别为 1.8% 和 2.5%。这样一来，如果两个广告出价相当，优先展示广告 B 似乎更合理些。

实际问题远没有这么简单。首先，这种办法对于新的广告显然不合适，因为它们没有被点击的历史数据。第二，即使对于旧的广告，绝大部分时候，一个查询对应的特定广告不过两三次的点击。这时候，统计的数据严重不足，很难说被点了三次的就比被点了两次的好；这就好比我们不能从楼上看到楼下有三个女生、两个男生，就得到这个城市的男女比例是 2∶3 一样。

第三，广告的点击量显然和它们摆放的位置有关：放在第一条的广告的点击率理所当然比第二条的点击率要高很多。因此，在做点击率预估时，必须消除这个噪音。最后还要指出，影响点击率的因素非常多，这些都是要在做点击率预估时考虑的。

现在麻烦了，这么多因素要用一个统一的数学模型来描述看来不是一件容易的事情。更何况我们还希望这个模型能够随着数据量的增加越做越准确。早期有很多对经验值进行修正和近似的做法，但是在整合各个特征时，效果都不是很好。后来工业界普遍采用了逻辑回归模型（Logistic Regression 或 Logistic Model）。

2　逻辑回归模型

逻辑回归模型是将一个事件出现的概率适应到一条逻辑曲线（Logistic Curve，其值域在 (0, 1) 之间）上。逻辑曲线是一条 S 型的曲线，其特点是开始变化快，逐渐减慢，最后饱和。比如一个简单的逻辑回归函数有如下形式

$$f(z)=\frac{e^z}{e^z+1}=\frac{1}{1+e^{-z}} \tag{28.1}$$

对应如下的曲线：

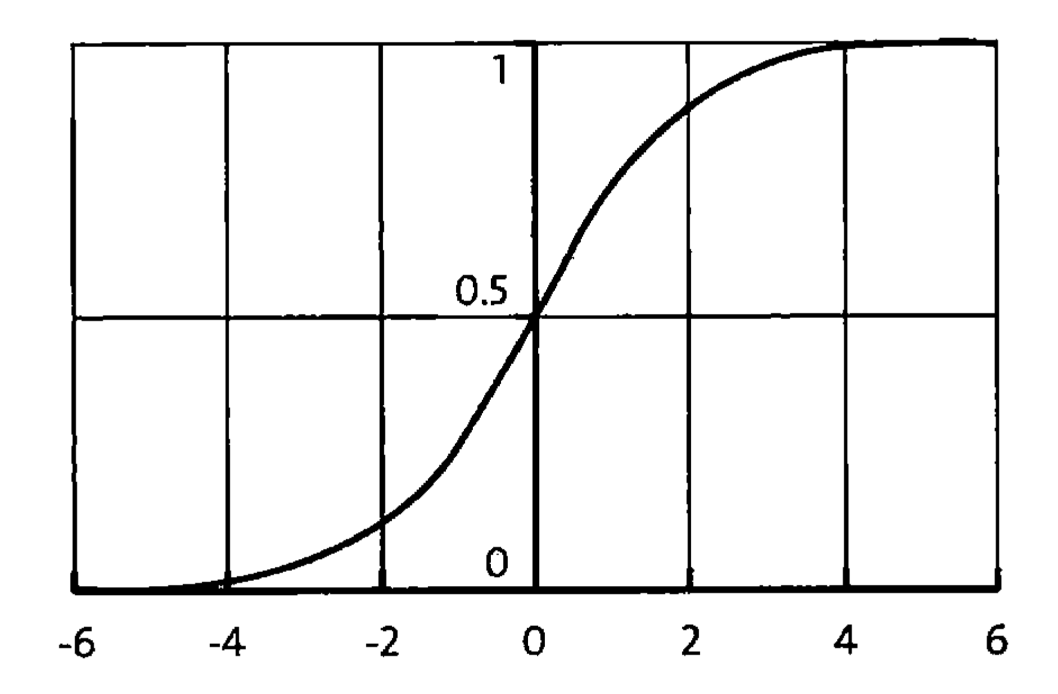


图 28.1　逻辑回归函数的曲线

逻辑自回归的好处是它的变量的范围是从 $-\infty$ 到 $+\infty$，而值域范围限制在 0-1 之间。（当然，由于 z 超出 [-6，6] 后函数值基本上没有变化，在应用中一般不考虑）我们知道对应于 [0, 1] 之间的函数可以是一个概率函数，这样逻辑回归函数就可以和一个概率分别联系起来了。而自变量 z 的值在（$-\infty$，$+\infty$）的好处是，它可以把这种信号组合起来，不论组合成多大或者多小的值，最后依然能得到一个概率分布。

回到上述预估点击率的问题，假如有 k 个影响点击率的变量 $x_1, x_2, \ldots, x_k$。用线性的办法将它们组合起来

$$z=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_kx_k \tag{28.2}$$

这里面，每一个 x_i 被称为变量，它们代表了影响概率预测的各种信息，比如广告的位置、广告和搜索词的相关性、广告展现的时间（比如晚上广告的点击率会略高于下午）。对应的 β_i 被称为其自回归参数，表示相应变量的重要性。β_0 是一个特殊的参数，和任何变量无关，可以保证在没有任何信息时，有一个稳定的概率分布。

下面看一个简单的例子。预测一下一个有关鲜花搜索的广告点击率。假定几个影响点击率的因子分别是每千次展示的点击次数（或者说单位点击量所需要的展现量）、广告和搜索的相关性（对应第二个变量 X_2）、目标人群的性别（对于第三个变量 X_3）等。

假定 X_1 对应单位点击所需的展现量，X_2 对应广告和搜索的相关性，在 0-1 之间，1 为完全匹配，0 为毫无关系，X_3 对应性别，1 为男性，0 为女性。

假定对应的参数 $\beta_0 = 5.0$，$\beta_1 = 1.1$，$\beta_2 = -10$，$\beta_3 = 1.5$。

比如搜索关键词是鲜花，广告是玫瑰，对应的变量值分别为 $X_1 = 50$，$X_2 = 0.95$，用户为男性：

那么 $Z = 5 + 1.1 \times 50 + (-10) \times 0.95 + 1.5 \times 1 = 52$，那么点击率的预估

$$P = \frac{1}{Z} = 0.019 = 1.9\% \qquad (28.3)$$

这里面的技巧有两点。第一是如何选取与广告点击相关的信息，这些是专门从事搜索广告的工程师和数据挖掘的专家的工作，这里就不赘述了。我们集中介绍一下第二点——如何决定这些参数 β。

上面的逻辑回归函数其实上是一个一层的人工神经网络，如果需要训练的参数数量不多，所有训练人工神经网络的方法都适用。但是，对于搜索广告的点击率预估这样的问题，需要训练的参数有上百万个，因此要求更有效的训练方法。读者可能已经发现，具有公式（28.1）形态的逻辑回归函数和我们在前面介绍过的最大熵函数，在函数值和形态上有着

共性，它们的训练方法也是类似的，训练最大熵模型的 IIS 方法可以直接用于训练逻辑回归函数的参数。

一个广告系统中，有没有很好的点击率预估机制决定是否能成倍提高单位搜索的广告收入。而目前 Google 和腾讯的广告系统对于点击率预估的方法，都是采用逻辑回归函数来预测的。

3 小结

逻辑回归模型是一种将影响概率的不同因素结合在一起的指数模型。和很多指数模型（例如最大熵模型）一样，它们的训练方法相似，都可以采用通用迭代算法 GIS 和改进的迭代算法 IIS 来实现。除了在信息处理中的应用，逻辑回归模型还广泛应用于生物统计。

第29章 各个击破算法和Google云计算的基础

云计算在七年前（2005年）被划了一个大大的问号，今天，连非IT行业的人都开始谈论这个问题。2011年，我至少参加了七八个云计算的研讨会、标准制定会。总的感觉是，大家对云计算的表层多有了解，但是，对技术的关键点了解甚少。

云计算技术涉及的面很广，从存储、计算、资源的调度到权限的管理等，有兴趣的读者可以参看拙作《浪潮之巅》中关于云计算的一章，这里不再赘述。云计算的一个关键问题是，如何把一个非常大的计算问题，自动分解到许多计算能力不是很强大的计算机上，共同完成。Google针对这个问题给出的解决工具是一个叫MapReduce的程序，其根本原理就是计算机算法上很常见的分治算法(Divide-and-Conquer),我称之为“各个击破”法。

1 分治算法的原理

分治算法是计算机科学中最漂亮的工具之一。它的基本原理是：将一个复杂的问题，分成若干个简单的子问题进行解决。然后，对子问题的结果进行合并，得到原有问题的解。

2 从分治算法到 MapReduce

熟悉计算机算法的读者都知道归并排序（Merge Sort）的原理。假定要对一个长度为N的数组$a_1,a_2,a_3,\cdots,a_N$进行排序，如果采用对a_i和a_j两两比较的办法（冒泡排序），复杂度是$O(N^2)$，不仅非常笨（慢），而且如果数组过大（比如几千亿个元素），也无法在一台计算机上完成。用分治算法，是将这个大数组分为几份，比如一分为二，变为$a_1,a_2,\cdots,a_{N/2}$和$a_{N/2+1},a_{N/2+2},\cdots,a_N$，对每一半分别进行排序。当这两个子数组排序完毕后，把它们从头到尾合并，得到原来数组的排序结果。对应的大小正好是原数组一半大，只需要进行 1/4 的比较即可。当然，合并的过程需要一些额外的时间，但是和省下的时间相比可以忽略不计。同理，还可以对前后每一半子数组继续分解成更小的子数组，直到子数组中只有两个元素。这种做法大大缩短了整个排序的时间，由原来的$O(N^2)$简化到$O(N\cdot\log N)$ [1]，如果N是一百万，那么计算时间将缩短一万倍。这个排序算法在每个子任务完成后，都要进行合并，归并排序算法也因此得名。

1 推导如下：假定N个元素的数组归并排序的算法计算时间是$T(N)$，那么$N/2$个元素的子数组排序时间为$T(N/2)$，合并过程的计算时间为N的线性函数，即$O(N)$。因此$T(N)=2T(N/2)+O(N)$。解这个递归方程，即得$T(N)=O(N\cdot\log N)$。

假定矩阵$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}$，同时矩阵$B=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N} \\ b_{21} & b_{22} & \cdots & b_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ b_{N1} & b & \cdots & b_{NN} \end{bmatrix}$，

要计算它们的乘积$C=A\times B$。

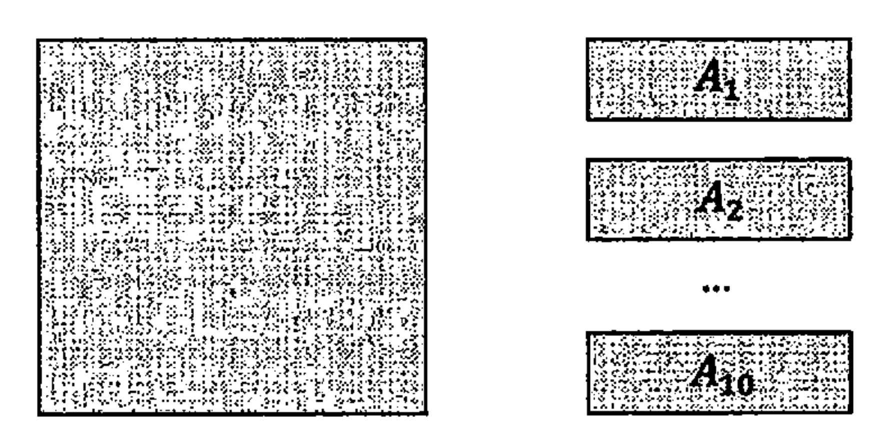
$$c_{nm}=\sum_i a_{ni}\cdot b_{im} \qquad (29.1)$$

做上面的计算要扫描矩阵A中n行的所有元素，以及矩阵B中m列的所有元素。

$$\begin{bmatrix} c_{11} & \cdots & c_{1j} & \cdots & c_{1N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{i1} & \cdots & c_{ij} & \cdots & c_{iN} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{N1} & \cdots & c_{N2} & \cdots & c_{NN} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{iN} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{N1} & \cdots & a_{Nj} & \cdots & a_{NN} \end{bmatrix} \times \begin{bmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{i1} & \cdots & b_{ij} & \cdots & b_{iN} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{N1} & \cdots & b_{Nj} & \cdots & b_{NN} \end{bmatrix} \quad (29.2)$$

如果一台服务器存不下整个大数组，这件事就变得很麻烦。好了，让我们看看分治算法怎么做。首先，把矩阵 A 按行拆成 10 个小矩阵 $A_1, A_2, \ldots, A_{10}$，每一个有 $N/10$ 行。



A：$N \times N$ 的矩阵　　$A_1, A_2, \ldots, A_{10}$：$N/10 \times N$ 的矩阵

图 29.1　将矩阵 A 按行分成 10 个子矩阵 $A_1, A_2, \ldots, A_{10}$

分别计算每个小矩阵 $A_1, A_2, \ldots, A_{10}$ 和 B 的乘积，不失一般性，以 A_1 来说明，

$$c_{nm}^1 = \sum_i a_{ni}^1 \cdot b_{im} \quad (29.3)$$

这样就在第一台计算机上计算出 C 矩阵中前 1／10 行的元素。

$$C_1 = A_1 \times B$$

图 29.2　第一台服务器完成前十分之一的计算量

同理，可以在第二台、第三台、……服务器上计算出其他元素。当然，细心的读者可能会发现，矩阵 B 和矩阵 A 一样大，一台服务器同样存不下。不过没有关系，同样可以按列切分矩阵 B，使得每台服务器只存矩阵 B 的 1／10。上述公式可以直接使用，只是这回只完成了 C_1 的 1／10。

因此，这次需要 100 台服务器而不是原来的 10 台，于是，在单机上无法求解的大问题就被分解成小问题得以解决。

$C_{1,5}$ = A_1 × B_5

图 29.3　第一台服务器的工作被分配到 10 台中，这是其中的第五台

在上面的例子中，增加了服务器的数量，但是每个元素 $c_{n,m}$ 的绝对计算时间其实并没有减少（这点不像归并排序）。但是在某些应用中，增加服务器数量可以带来绝对计算时间的减少。例如，只是要得到 C 中某个特定元素 $c_{n,m}$ 而不是整个 C 矩阵，这种需求在链接分析或者日志处理中有时会遇到，假如希望通过 10 倍的机器能够缩短计算时间。这个需求也可以通过分治算法来实现，具体解决方法如下：

对矩阵 A 按行切分，对矩阵 B 按列切分。A_1 是 A 的前十分之一行，A_2 是接下来的十分之一，等等，对 B 也是如此。然后用同样的方法计算得到 10 个中间结果 $c_{nm}^1, \ldots, c_{nm}^{10}$。而 $c_{n,m}$ 只是这 10 个数字相加的结果。下面每个结果的计算量都是最后结果的十分之一。这样，我们就用 10 倍的计算机数量将计算时间缩短了 10 倍。

$$
\begin{aligned}
c_{nm}^1 &= \sum_{i=1}^{\frac{N}{10}} a_{ni} \cdot b_{im} \\
c_{nm}^2 &= \sum_{i=N/10+1}^{\frac{2N}{10}} a_{ni} \cdot b_{im} \\
&\cdots\cdots \\
c_{nm}^{10} &= \sum_{i=9N/10+1}^{N} a_{ni} \cdot b_{im}
\end{aligned}
\qquad (29.4)
$$

这就是 MapReduce 的根本原理。将一个大任务拆分成小的子任务，并且完成子任务的计算，这个过程叫做 Map，将中间结果合并成最终结果，这个过程叫做 Reduce。当然，如何将一个大矩阵自动拆分，保证各个服务器负载均衡，如何合并返回值，就是 MapReduce 在工程上所做的事情了。

在 Google 开发 MapReduce 之前，上述思想已经在很多高强度计算上使用了。我在约翰·霍普金斯大学训练最大熵模型时就遇到过类似的问题。我经常需要用 20 台左右的服务器（这在 Google 云计算出来前已经是非常奢侈了）同时工作。那么我工作的方式就是手工将这些大矩阵拆开并且推送（Push）到不同的服务器上，然后把结果组合起来。伯克利加州大学提供了一个在操作系统上检测各个子任务完成情况的工具，这样，我需要自己写一个批处理流水线来完成 Map 和 Reduce 这两个过程。唯一的差别是，有了 MapReduce 工具，所有的调度工作都是自动完成的，而在此以前，我是自己完成计算机的调度工作。

3 小结

我们现在发现 Google 颇为神秘的云计算中最重要的 MapReduce 工具，其实原理就是计算机算法中常用的“各个击破”法，它的原理原来这么简单 —— 将复杂的大问题分解成很多小问题分别求解，然后再把小问题的解合并成原始问题的解。由此可见，在生活中大量用到的，真正有用的方法往往简单而又朴实。

附录　计算复杂度

计算机算法的效率是用计算复杂度（Computational Complexity）来衡量的。计算的时间显然和问题的大小有关。比如对 10 000 个实数的排序和对 1 000 000 个实数的排序，所用时间显然不同。因此，问题的大小在衡量计算复杂度时是变量，一般用 N 来表示。而计算量是 N 的一个函数 $f(N)$。这个函数的边界可以用数学上的大 O 概念来限制。如果两个函数 $f(N)$ 和 $g(N)$ 在大 O 概念上相同，也就是说当 N 趋近于无穷大时，它们的比值只差一个常数。比如 $f(N) = N \cdot \log N, g(N) = 100 \cdot N \cdot \log(N)$，它们就被看成是同一数量级的。同样，如果两个计算机算法的计算在大 O 概念下相同，只相差一个常数，我们认为它们的计算复杂度相同。

计算的复杂度关键看 $O()$ 里面的函数，而不是常数。比如假定两个算法的计算量分别是 $10000 \cdot N \cdot \log(N)$ 和 $0.00001 \cdot N^2$，虽然前者的常数大得多，但是当 N 趋近于无穷大时，后者的计算量是前者的无穷大倍。对于这两个算法，我们把它们的计算复杂度分别写作 $O(N\log N)$ 和 $O(N^2)$。

如果一个算法的计算量是 N 的多项式函数（Polynomial Function），则

认为这个算法是多项式函数复杂度的。如果一个问题存在一个多项式函数复杂度的算法，则这个问题称为 P 问题（Polynomial 的首字母）。如果计算量比N的多项式函数还高，虽然从理论上讲如果有无限的时间也是可以计算的（图灵机概念下的可计算），但是实际上是不可计算的。这时称它为非多项式（Non-deterministic polynomial，简称 NP）问题。比如找到每一步围棋的最佳走法就是 NP 问题。

如果一个 NP 问题，虽然它找不到多项式函数复杂度的算法，但是对于一个算法可以在多项式函数复杂度的时间里证实这个方法正确与否，那么这个问题成为 NP-Complete 问题。如果一个问题，它的计算复杂度至少是 NP-Complete，那么它被称为 NP-Hard 问题，NP-Complete 和 NP-Hard 的关系如下。

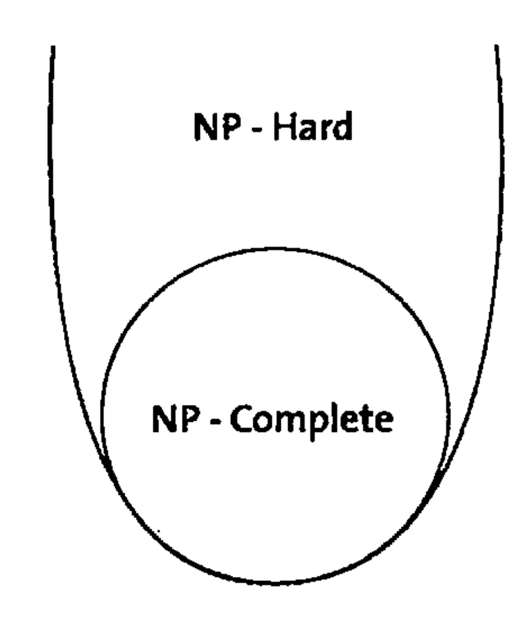


图 A.1 NP-Complete 问题是 NP-Hard 问题的特例

后记

很多朋友问我，为什么我会想起来写这个系列？虽然谷歌黑板报的本意是希望我从一个 Google 科学家的角度介绍一下 Google 的技术，但是我更希望让做工程的年轻人看到在信息技术行业正确的做事情方法。无论是在美国还是在中国，我经常看到大部分软件工程师在一个未知领域都是从直观感觉出发，用“凑”的方法来解决问题，在中国尤其如此。这样的做法说得不好听，就是山寨。我刚到 Google 时，发现 Google 早期的一些算法（比如拼写纠错）根本没有系统的模型和理论基础，就是用的词组或者词的二元组凑出来的。这些方法比没有做任何事情是好一些，但是几乎没有完善和提高的可能，而且使得程序的逻辑非常混乱。Google 成长壮大后，渐渐有实力从世界上最好的大学招理论基础非常好的工程师，工程的正确性得到了很好保证。2006 年后，我指导了三四个美国名校的研究生，把 Google 的拼写纠错模型用隐含马尔可夫模型的框架统一起来。在那几年里，Google 对几乎所有项目的程序进行了重写，山寨的东西基本上看不到了。但是在其它公司里，包括在美国一些还挂着高科技头衔的二流 IT 公司里，这种情况依然很普遍。在国内，创业的小公司做事情重量不重质，倒也无可厚非；但是，上了市、有了钱甚至利润成为了在

世界上也数得上的公司，做事情依然如此，就让人觉得境界低。另一方面，这些公司在盖大楼和装修高管的办公室上很快超越了世界上的跨国公司。这就像一个人有了钱，穿金戴银，内在的学问和修养却没有提高一样。因此我写这些东西也是希望我们这些 IT 公司的工程主管们能够带领自己的部门提高工程的水平。

（无意中）采用错误的模型在特定的场合，或许勉强有效，就比如我们介绍的地心说一样，毕竟也使用了几千年。但是，错误的模型终究是远离真理的，其负面影响会渐渐表现出来。其结果不仅仅在于远离了正确的结果，而且常常把原本简单的事情弄得很复杂，以至于最终要崩溃（地心说对于日心说就是如此）。

正确的理论和方法有一个被认识的过程。任何事物都有它的发展规律，而这些规律都是可以认识的，在信息科学领域也不例外。当我们认识了规律后，就应该自觉地在工作中遵循规律而不要违背规律。香农博士就是揭示了信息科学发展规律的人，它的信息论在很大程度上指出了我们今天信息处理和通信根本的规律性。这里，通信包括人类的一切交流，包括自然语言处理的所有应用。而当初我写这个系列博客，就是要介绍这些信息处理的规律性。

当然，将数学的东西讲清楚让外行都能读懂是一件非常难的事情。我自认为自己是一个能深入浅出的人，但是当我第一次将所写的几章送给非工程专业的读者阅读时，他们还是表示非常费劲。因此，我后来下了很多功夫将这个系列写得浅显易懂，这样很多细节只能省略，我并不满意。离开 Google 后，写作起来约束相对少了些，因此这次改写成实体书时，可以多介绍一些细节。同时，由于篇幅不受约束，我也可以多提供一些细节，以照顾一下工程背景较好的、愿意了解细节的读者。当我完成这本实体书时，我发现全书的内容完全重写了一遍。

对于非 IT 的从业人员，我也希望这本书能够成为他们茶余饭后消遣的科普读物。透过对 IT 规律性的认识，读者可以举一反三地总结、学习、认

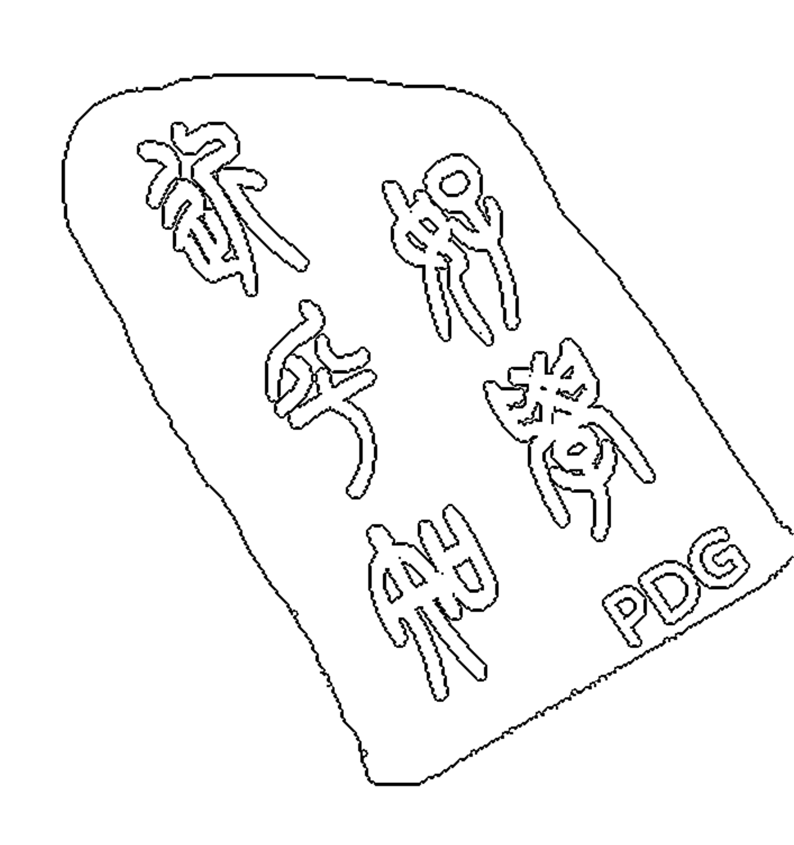
识和自觉使用自己工作中的规律性，这样有助于将自己的境界提升一个层次。

对我这次写作帮助最大的是两本书和一个节目。我在初中时读了《从1到无穷大》[1]，介绍宇宙的科普读物。作者G·伽莫夫（George Gamow）是美籍俄裔著名物理学家，他花了很多时间创作科普读物，影响了一代人。第二本书是物理学家霍金的《时间简史》，霍金把深奥的宇宙学原理用最简单的语言讲出来，让这部科普读物称为全球的畅销书。影响我的一个节目是美国主持人摩根·弗里曼的“穿越虫洞”。我的写作大多是在飞机上完成的，写作累了便看看电视节目，一次碰巧找到“穿越虫洞”这个节目。弗里曼把当今最前沿的物理学做成了用每个人都能懂的节目。节目中有包括很多诺贝尔奖在内的一流物理学家和数学家介绍他们的工作，这些人有一个共同的本领，就是把他们自己领域最深奥的道理用很简单的比喻介绍清楚。我想这可能是他们成为世界顶级科学家的原因，他们一方面对自己的领域非常精通，同时他们能把道理讲清楚。世界上最好的学者总是可以深入浅出把大道理讲给外行听，而不是故弄玄虚把简单的问题复杂化。因此，在写这本书的时候，我自己一直以霍金、伽莫夫为榜样，力图将数学之美展现给所有的，而不仅仅是专业的读者。为了方便读者利用茶前饭后的时间阅读，我尽可能地做到每一章之间相对独立自成一体，这样读起来不会太累，我知道让大部分读者从头到尾读一本以数学为主的书是几乎不可能的。

1 原书名“One,Two,Three,…,Infinity”

吴军

2012年4月于深圳


PDG

索 引

A

B

C

D

E

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

X

Y

Z